8-8-2017

Regularized Numerical Algorithms For Stable Parameter Estimation In Epidemiology And Implications For Forecasting

Linda DeCamp

REGULARIZED NUMERICAL ALGORITHMS FOR STABLE PARAMETER ESTIMATION IN EPIDEMIOLOGY AND IMPLICATIONS FOR FORECASTING

by

LINDA DECAMP

Under the Direction of Professor Alexandra Smirnova

ABSTRACT

When an emerging outbreak occurs, stable parameter estimation and reliable projections of future incidence cases using limited (early) data can play an important role in optimal allocation of resources and in the development of effective public health intervention programs. However, the inverse parameter identification problem is ill-posed and cannot be solved with classical tools of computational mathematics. In this dissertation, various regularization methods are employed to incorporate stability in parameter estimation algo-

rithms. The recovered parameters are then used to generate future incident curves as well as the carrying capacity of the epidemic and the turning point of the outbreak.

For the nonlinear generalized Richards model of disease progression, we develop a novel iteratively regularized Gauss-Newton-type algorithm to reconstruct major characteristics of an emerging infection. This problem-oriented numerical scheme takes full advantage of *a priori* information available for our specific application in order to stabilize the iterative process. Another important aspect of our research is a reliable estimation of time-dependent transmission rate in a compartmental SEIR disease model. To that end, the ODE-constrained minimization problem is reduced to a linear Volterra integral equation of the first kind, and a combination of regularizing filters is employed to approximate the unknown transmission parameter in a stable manner. To justify our theoretical findings, extensive numerical experiments have been conducted with both synthetic and real data for various infectious diseases.

INDEX WORDS: Inverse Problems, Epidemiology, Regularization, Parameter Estimation, Forecasting

REGULARIZED NUMERICAL ALGORITHMS FOR STABLE PARAMETER ESTIMATION IN EPIDEMIOLOGY AND IMPLICATIONS FOR FORECASTING

by

LINDA DECAMP

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2017

REGULARIZED NUMERICAL ALGORITHMS FOR STABLE PARAMETER ESTIMATION IN EPIDEMIOLOGY AND IMPLICATIONS FOR FORECASTING

by

LINDA DECAMP

Committee Chair: Alexandra Smirnova

Committee: Vladimir Bondarenko

Gerardo Chowell-Puente

Michael Stewart

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

June 2017

DEDICATION

For

Michael

ACKNOWLEDGEMENTS

Above all, my work in research would not have been possible without the direction and support of my advisor, Dr. Alexandra Smirnova. Her knowledge base is astounding and over the years I have had the opportunity to learn a great deal with great depth and I am privileged to have worked with her. Over the years, Dr. Smirnova has provided so many avenues to improve as a scholar, researcher and instructor; her belief in me has meant a great deal to me. Thank you for your patience, understanding and continued confidence in me and my abilities; with these and more this effort has been successful.

The success of this work would also not have been possible without the advice, support and feedback of my committee members, Dr. Vladimir Bondarenko, Dr. Gerardo Chowell and Dr. Michael Stewart. Your time and efforts on my behalf were much appreciated. I have had the opportunity to work with each of you before, and the aggregate of our combined previous achievements made this process that much more enjoyable and productive.

I must also gratefully acknowledge Dr. Valerie Miller and Dr. Rebecca Rizzo, both of whom I had as instructors in my first semester back at Georgia State fulfilling post-baccalaureate course requirements. It was Dr. Miller who persuaded me to apply to the graduate program. In the ensuing years, she has been a helpful, honest and forthright resource and sounding board in all matters academic; her direct clarity benefited me on many occasions. Dr. Rizzo has been another constant, friendly and outgoing in a way that both encouraged and acknowledged me in whatever association we found ourselves. I was able to work directly with Dr. Rizzo as she pursued different lab and course development planning and implementation, an invaluable experience.

In attempting to list those that in addition I wish to acknowledge, I find that this includes the vast majority of the math department. Over the years I have been able to take classes and/or work with so many of you and can only say that I truly appreciate all that you have given to me. For my support among my cohorts, I do want to recognize my

predecessors Dr. Hui Liu and Dr. Leslie Meadows; you both helped me develop and improve. By showing me the way, you also made my way easier to walk. To Aurelie Akossi, I wish you all the best. Once again, considering my fellow graduate students, there are additionally too many to name; thank you all.

This dissertation is dedicated to my husband, Michael, who has been my biggest fan and supporter. There truly is not enough space to acknowledge all he has done. He is joined by my children who also have cheered me on, being there for me in ways both small and large. Thank you Danny, Elizabeth, Jacob, Eli, Samantha and Ty. I hope that I can support each of you in your dreams as well as you have supported me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- PDE - Partial Differential Equation
- TSVD - Truncated Singular Value Decomposition
- IRGN - Iteratively Regularized Gauss-Newton
- ODE - Ordinary Differential Equation
- SVD - Singular Value Decomposition
- EVD - Ebola Virus Disease
- WHO - World Health Organization
- RIRGN - Reduced Iteratively Regularized Gauss-Newton
- LSP - Least Squares Problem
- BVP - Boundary Value Problem
- IVP - Initial Value Problem
- CI - Confidence Interval
- SIR - Susceptible-Infected-Recovered
- MCMC - Markov Chain Monte Carlo
- MTSVD - Modified Truncated Singular Value Decomposition
- SEIR - Susceptible-Exposed-Infected-Recovered

PART 1

INVERSE PROBLEMS

1.1 Ill-posed Problems

The concept of a well posed problem in Partial Differential Equations (PDEs) goes back to J. Hadamard [2], and it had been introduced in the attempt to determine what types of boundary conditions are most suitable for various types of differential equations (for example, the Dirichlet boundary conditions for elliptic equations and the Cauchy boundary conditions for hyperbolic equations). Since the pioneer work of J. Hadamard, the notion of a well-posed problem has been generalized to an arbitrary operator equation, $A(x) = f$ on a pair of metric spaces X and Y with the metrics ρ_X and ρ_Y, repectively. The formal definition can be stated as follows.

Definition. The problem of finding a solution x in X from the data f in Y, $A(x) = f$, is said to be (Hadamard) *well-posed* if the following conditions are satisfied:

a) for every element $f \in Y$ there exists a solution $x \in X$,

b) this solution is unique,

c) the problem is stable under perturbations of the input data, f.

Otherwise the problem is *ill-posed*. Special methods are to be used for solving such problems. Numerous examples of ill-posed problems in pure mathematics and in real-life applications can be found in [3–7]. Among classical ill-posed problems are Fredholm and Volterra integral operator equations of the first kind, numerical differentiation of noisy data, inversion of ill-conditioned matrices, summation of Fourier series (and integrals) with noisy coefficients, unconstrained minimization of non-convex functionals, and many others. Ill-posed problems arise in astrophysics, geophysics, ocean acoustics, spectroscopy, computerized tomography and other areas of sicence and engineering.

Suppose that the problem of solving the equation

$$A(x) = f, \quad A : X \to Y, \tag{1.1}$$

is ill-posed, and the right-hand side f is given by its $\delta-$approximation f_δ such that $\rho_Y(f, f_\delta) \leq \delta$. It is natural to seek an approximate solution of the equation (1.1) in the class $Q_\delta := \{x \in X : \rho_Y(Ax, f_\delta) \leq \delta\}$. However in the ill-posed case an arbitrary element $x_\delta \in Q_\delta$ cannot be taken as an approximate solution to (1.1), since $\rho_X(x, x_\delta)$ does not go to 0 as δ goes to 0, in general. In order to select a suitable solution one needs to use *a priori* information (usually available) about x, which may be of a quantitative or qualitative nature.

The usage of quantitative *a priori* information makes it possible to narrow the class of solutions, for example, to a compact set, M, so that the problem becomes stable under small changes in the input data. This leads to a concept of a *quasisolution* introduced by V. Ivanov in [8, 9]. Various algorithms for approximate computation of quasisolutions and various combinations of assumptions on A, Y, and M that guarantee the well-posedness of the corresponding quasisolution problem were studied in [9–12].

A priori information of a qualitative nature (for example, sparsity or smoothness of the solution) generates different approaches. The most known amoung them is *variational Tikhonov's regularization* [13, 14], which allows one to construct stable approximate solutions to ill-posed problems by means of a stabilizing functional. The variational method has been extensively developed in [6, 7, 15, 16], and certain *a priori* and *a posteriori* choices of regulariztion parameter $\alpha = \alpha(\delta)$ have been designed and implemented [17–23].

One can also find approximate solutions to (1.1) by iterations (see [5, 24–26]), taking $x_n = Z(f_\delta, x_{n-1}, \cdots, x_{n-k})$, where $k \leq n$. For these solutions to be stable under small changes in the initial data, the iteration number $n = n(\delta)$ yielding x_n must be a function of the size of the error in the initial data.

Other important algorithms in the theory of ill-posed problems include Lavrentiev's

(shift) regularization [3], local regularization [27], truncated singular value decomposition (TSVD) [6], various procedures for solving inverse scattering problems [28, 29], the level set methods [30], and regularization parameter selection methods based on prior statistical information [31–36].

1.2 Parameter Estimation Inverse Problems

Parameter estimation problems in ordinary and partial differential equations constitute a large class of models described by ill-posed operator equations. Here one is trying to identify the coefficients of a differential equation, called system parameters, from observations of the solution to that equation. An extra level of difficulty is added by the fact that even when the differential equation is linear with respect to the solutions and its derivatives, the corresponding inverse (parameter estimation) problem is generally nonlinear. If this nonlinear problem is solved with some Newton-type iterative algorithm, an ill-posed linear operator equation in the form (1.1) needs to be solved at every step of the iterative process. Due to instability, the error accumulates and completely destroys the iterative solution. Thus, for parameter estimation problems, the regularization component of a numerical algorithm becomes highly important.

A considerable number of parameter identification problems come from epidemiology and infectious disease modeling. These problems have some unique challenges.

1. In the past years, models developed with annualized data primarily used constant system parameters [37–41]. For a relatively small number of constant parameters, the corresponding optimization problem is likely to be over-determined and relatively stable. In this case, a pre-packaged optimization routine can be successfully employed. Matlab provides a number of such routines (for example, lsqcurvefit, lsqnonlin, and others). However, with the advent of more timely and frequent reporting of clinical data, some system parameters (like bird-to-human transmission rate in avian influenza) emerge as time dependent due to seasonality and other environmental factors. With variable system parameters, the dimensionality of the solution space is growing and

the optimization problem becomes under-determined. In order to solve it in a stable fashion, a rather sophisticated "problem-oriented" regularizing algorithm must be proposed.

2. One of the main dangers of instability is that we can get a very good fit but with highly inaccurate system parameters. Thus, regularization is essential, and one has to find a regularization parameter, α. However, the noise level in clinical data is almost impossible to estimate. For this reason, one has to resort to heuristic methods for the evaluation of α. Generally, these methods provide us with some insights into how α can be selected, but the choice is not theoretically justified. As a safety net, it is desirable to obtain consistent values of the regularization parameter with more than one heuristic method in order to construct an effective stabilizing algorithm.

3. Some infectious diseases are modeled by systems of differential equations that include humans and other species (domestic poultry, mosquitos, etc.). Since human data is usually more reliable, non-human parameters (such as bird-to-bird transmission rate in avian influenza or mosquito transmission coefficient in *Plasmodium falciparum* malaria) need to be fitted to human data (cumulative number of human H5N1 cases or population size with clinical malaria symptoms, for example) [42, 43]. The corresponding optimization problem turns out to be highly nonlinear and, in case of variable parameters, very unstable. Once again, special regularization algorithms must be incorporated into numerical optimization schemes for a successful estimation of system parameters.

4. The inherently differing scales of biological parameters in a disease model may complicate their simultaneous recovery by a regularized optimization algorithm based on the original Tikhonov functional with $\mathcal{L}_2$ penalty term. For example, in the case of H5N1 virus, bird-to-human transmission rate, $\beta(t)$, is of order 10^{-8} or 10^{-7}, while bird-to-bird transmission rate, $\beta_b(t)$, is of order 10^{-3} [37, 39, 41, 43] (data related to humans and poultry is in units 10^5 and 10^7 individuals, respectively; time is in months). With these two parameters being 5 or 4 orders of magnitude apart, the sensitivities of the cost

functional with respect to each variable will also be on different scales. This suggests that the penalty term on $\beta(t)$ should be appropriately weighted (say, through a more general Tikhonov's penalty term, $||Lx||^2$) to ensure convergence in both variables.

In Chapter 2, we study a parameter identification problem involving a nonlinear differential equation, where the primary goal is to recover the carrying capacity of the outbreak and the incidence turning point. As mentioned, nonlinear inverse problems require an iterative algorithm with an intrinsic regularization procedure, and the Iteratively Regularized Gauss-Newton (IRGN) scheme [44] can be viewed as a reliable starting point. Using the IRGN method as a foundation, we develop a special problem-oriented algorithm wherein modifications are made to the original disease model as well as to the iterative scheme through the reduction of the Jacobian and the introduction of a weighted penalty operator addressing differences in scales of system parameters. This gives rise to what we call the Reduced IRGN (RIRGN) scheme. Numerical experiments are presented to illustrate advantages and limitations of the proposed algorithm.

In Chapter 3, we focus on estimating the time-dependent transmission rate that produces incidence data in a disease dynamical system. The transmission rate of an infection is dependent on both contact rates and the probability of transmission between susceptible and infected individuals, neither which are easy or even possible to measure. Efficient and stable recovery of disease transmission rates by solving the underlying inverse problems results in advantageous opportunities to more accurately forecast incidence cases, hypothesize and test control strategies, and proactively inform agencies and the public about the progression of an outbreak. We propose a stable regularization algorithm based on the reduction of our original ODE model to a linear Volterra integral equation. Experiments with a number of data sets utilizing various SVD filters are carried out to illustrate the benefits and effectiveness of recovering disease transmission rates from limited early data.

PART 2

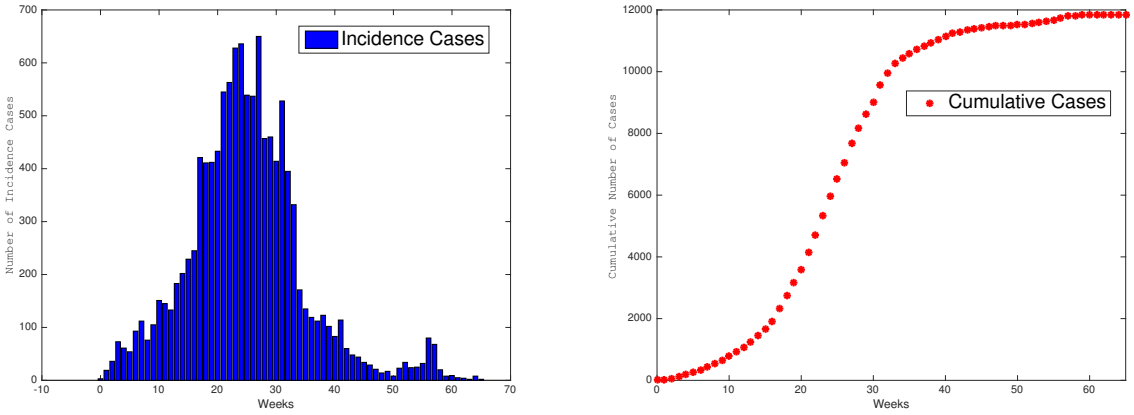
EARLY OUTBREAK PARAMETER RECOVERY

2.1 Introduction

One method of studying disease dynamics is the use of compartmental models and their associated systems of differential equations. This may be a reasonable approach when the goal is to recover various transmission rates, reproductive ratios or other related parameters. Alternative models rely on a direct representation of incidence and cumulative case curves resulting from actual data. These models are particularly beneficial in case of emerging diseases where the principal goal is to quantify the most significant parameters describing the nature of an impending epidemic. This may require, for example, fitting model predictions to a short-term data set comprised of aggregated time series of case incidence. Another crucial question is to understand how soon after the emergence of a new disease the key parameters of the outbreak can be projected. These parameters can guide effective allocation of resources that would reduce the impact of the outbreak's progression.

In Figure 2.1 we show representative incidence and cumulative case curves obtained from case data for the Ebola Virus Disease outbreak of 2014/15 in Sierra Leone [1]. The S-shaped cumulative case curve suggests a logistic-type model. From this model, given limited early data, important disease parameters such as the epidemic turning point and its overall capacity may be recovered. In this chapter, the algorithm for stable recovery of these parameters is presented and computational stability of the proposed regularization method is justified. The main results of this chapter have been published in [45].

We will conduct our numerical experiments using incidence data for the most recent outbreak of Ebola Virus Disease (EVD) in West Africa, predominately affecting Guinea, Liberia, and Sierra Leone [1]. This EVD outbreak, which began in early 2014, has received wide attention due to its scale, scope, location and alarming potential. The largest previ-



(a) *Incidence Cases* (b) *Cumulative Case Curve*

Figure 2.1. *Incidence and Cumulative Case Curves from Sierra Leone 2014-15 Ebola Outbreak*

ous Ebola outbreak was in Uganda in 2000, with a total of 425 cases. The West African outbreak surpassed the size of that outbreak by the first week of June, 2014. The World Health Organization (WHO) declared the latest Ebola outbreak a public health emergency on August 8th, 2014 [46]. By the 21st of that month the case count exceeded the total of all other previous outbreaks combined - 2,387 cases. As of the most recent WHO situation report (March 30th, 2016) there have been 28,646 Ebola cases with 11,323 fatalities [47], and these numbers are widely believed to be underreported.

Human-to-human EVD transmission results from direct contact through broken skin or mucous membranes with the blood and other bodily fluids of infected people. The incubation period, or the time interval from infection to onset of symptoms, is from 2 to 21 days. The patients become contagious once they begin to show symptoms [48]. They are not contagious during the incubation period. Individuals remain infectious as long as their blood and secretions contain the virus [49, 50]. Additionally, humans get infected from improperly handled corpses of infected individuals. The EVD data are notoriously noisy due to substantial under-reporting and differing reporting periods. This data provide a unique opportunity to investigate efficiency and stability of parameter identification algorithms.

The logistic model was originally developed by Verhulst for population dynamics [51]

and was later applied to a similar behavior of disease case data

$$\frac{dC}{dt} = rC\left[1 - \frac{C}{K}\right], \quad C(0) = C_0. \tag{2.1}$$

Here $C(t)$ is the cumulative outbreak size in question at time t, r is the intrinsic growth rate, and K is the carrying capacity of the infection. As total cumulative cases, $C(t)$, continue to grow, $C(t)/K$ approaches 1 and the incidence rate, $\frac{dC}{dt}$, decreases to 0 as the outbreak capacity is achieved. The solution to (2.1) is

$$C(t) = \frac{KC_0}{C_0 + (K - C_0)e^{-rt}}, \tag{2.2}$$

where $C(t) \to K$ as $t \to \infty$.

Initial experiments were undertaken to recover disease capacity, K, utilizing the logistic model, (2.1). Given early data for weekly cumulative cases, $\mathbf{D} = [D_1, D_2, \cdots, D_m]$ for the 2014/15 Ebola Virus Disease Outbreak in Sierra Leone, Liberia and Guinea, the values of r and K have been obtained using Matlab's least squares curve fit function, lsqcurvefit. It has been discovered that, with the exception of Liberia, the carrying capacity parameter, K, rises as data is added to the model. Where the value of K levels out, if it does, it occurs after the apex of the incidence curve for all data sets. Neither parameter stabilizes until well after epidemic peak. In addition, reconstructed cumulative case curves significantly deviate from actual data curves for both early and late time periods. It appears the two parameter model lacks the ability to either effectively reconstruct the data curve or recover parameter values early enough to be of use.

In 1959, Richards [52] proposed the following generalization of the logistic model to quantify growth of biological populations

$$\frac{dC}{dt} = rC\left[1 - \left(\frac{C}{K}\right)^a\right], \quad C(0) = C_0. \tag{2.3}$$

The addition of the parameter a allows the modification of the logistic curve to account for

deviation from the S-shaped dynamics of the standard logistic behavior; it accommodates an asymmetrical growth curve [53]. Just like the logistic curve, the Richards model implies that there is a single incidence case peak corresponding to the inflection point of cumulative case curve. The analytic solution to (2.3) is given by

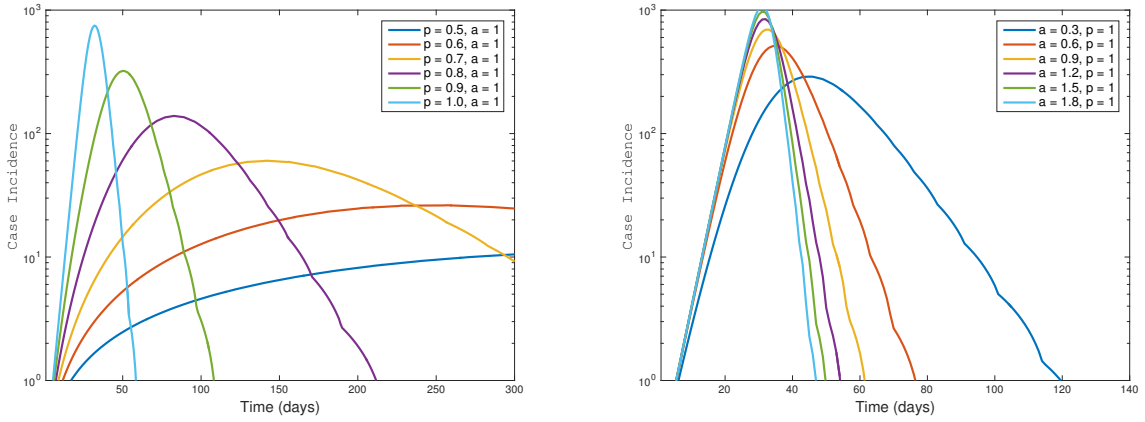
$$C(t) = \frac{KC_0}{(C_0^a + (K^a - C_0^a)e^{-art})^{(1/a)}}. \tag{2.4}$$

Implementation of parameter recovery with Richards model (2.3) exhibits similar results to the logistic model (2.1). Parameter values do not stabilize until well after the epidemic peak and although the case curve fit is slightly improved, significant deviations in early data persist. As in the logistic model (2.1), for Richards model (2.3) the early growth is minimally affected by the capacity of the outbreak. Indeed, in the early stages of an outbreak $(C(t)/K)^a$ is small and $\frac{dC}{dt} \approx rC$ resulting in exponential growth. However, previous investigations have indicated that the growth rate in early epidemics [54–57] is often sub-exponential. Several mechanisms could give rise to initial sub-exponential growth in case incidence including (i) spatially constrained contact structures, (ii) population behavioral changes and control interventions, and (iii) substantial heterogeneity in susceptibility and infectivity of the host population that can significantly distort the local structure of social contacts. To model such deceleration of growth, we introduce an additional parameter p and define the generalized Richards model [58]

$$\frac{dC}{dt} = rC^p \left[1 - \left(\frac{C}{K}\right)^a\right], \quad C(0) = C_0, \tag{2.5}$$

where r is the intrinsic growth rate, a measures the extent of deviation from the S-shaped dynamics of the classical logistic growth model [53], and K represents the epidemic final size, defined as the total number of infections throughout the epidemic. When $p = 1$ in (2.5) we have Richards model (2.3) with analytic solution (2.4). However, if $p \neq 1$, (2.5) has no closed form solution and must be solved numerically, although its analytic solution may be expressed in the form of an infinite series [53]. At the early stages of the epidemic,

this model allows the capture of different growth profiles ranging from constant incidence ($p = 0$), polynomial (or sub-exponential [55]) growth ($0 < p < 1$), to exponential growth ($p = 1$). Figure 2.2 illustrates a diversity of epidemic profiles, $\frac{dC}{dt}$, that the generalized Richards model supports, as p and a are varied.



(a) *Vary p, keep* $a = 1$ (b) *Vary a, keep* $p = 1$

Figure 2.2. *Simulated Incidence Curves from Varying Combinations of p and a where* $C(0) = 1$, $r = 0.3$, $K = 10,000$

The maximum incidence in the generalized Richards model takes the following form

$$\frac{dC}{dt}(\tau) = \frac{raK^p}{p}\left(\frac{p}{a+p}\right)^{1+\frac{p}{a}},$$

and the estimation of the time, τ, at which this maximum occurs for an emerging outbreak provides important information on the time-window available to implement the necessary intervention policies to reduce the number of infections. Past the peak-time of the epidemic, public health measures may have little effect on reducing the epidemic final size. In this chapter, we will develop a regularized numerical algorithm for estimating the inflection point and the epidemic final size using the generalized Richards model.

The rest of the chapter is organized as follows. In Section 2.2, the least squares problem with respect to parameters r, p, a, and K, is discussed and the lack of stability in the reconstruction of K is highlighted. In Section 2.3, the problem is reformulated in a more

stable manner with the unknown parameters having closer levels of magnitude. Advantages and limitations of the new formulation in case of both least-squares curve fitting trust-region algorithm in Matlab and our own implementation of iteratively regularized Gauss-Newton solver are presented. Further analysis of the optimization algorithm is proposed in Section 2.4. It is followed by the introduction of the Reduced Iteratively Regularized Gauss-Newton (RIRGN) method and numerical simulations demonstrating its efficiency in Section 2.5. The convergence analysis of the RIRGN is carried out in Section 2.6. Finally, in Section 2.7 we outline conclusions and directions for future work.

2.2 The least squares problem

In this section we use the most natural formulation of the inverse problem aimed at the recovery of parameters r, p, a, and K in equation (2.5). Given early cumulative data for a particular outbreak, $\mathbf{D} = [D_1, D_2, ..., D_m]$, we obtain a numerical solution, $C = C(r, p, a, K)$, to the initial value problem

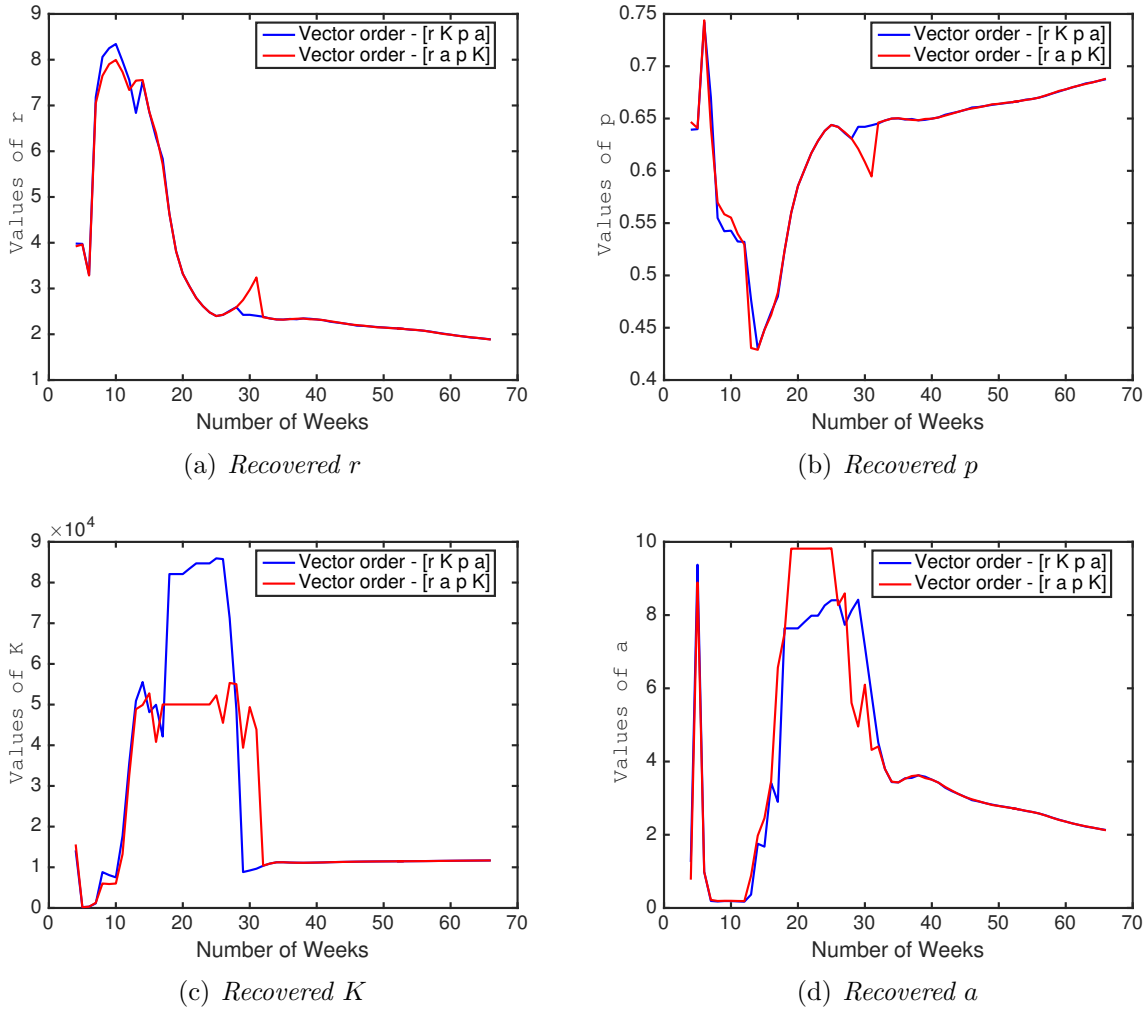
$$\frac{dC}{dt} = rC^p \left[1 - \left(\frac{C}{K}\right)^a\right], \quad C(t_1) = D_1,$$

at the same points $\{t_1, t_2, ..., t_m\}$ where the data are given. Optimizing the values of the unknown parameters to fit the corresponding data, we now have the following non-linear least squares problem

$$\min_{r,p,a,K} \frac{1}{2} \left\|C(r, p, a, K) - D\right\|^2 \quad C : \mathbb{R}^4 \to \mathbb{R}^m. \tag{2.6}$$

The simulation results hint to a substantial noise propagation in the reconstructed values of K prior to the inflection point, which undermines their reliability. We illustrate this phenomenon using cumulative data from the EVD outbreak in Sierra Leone.

In Figure 2.3, the impact of programming differences on the approximate values of system parameters is illustrated. Two codes use the same initial values: $r = 1$, $p = 1$, $K =$

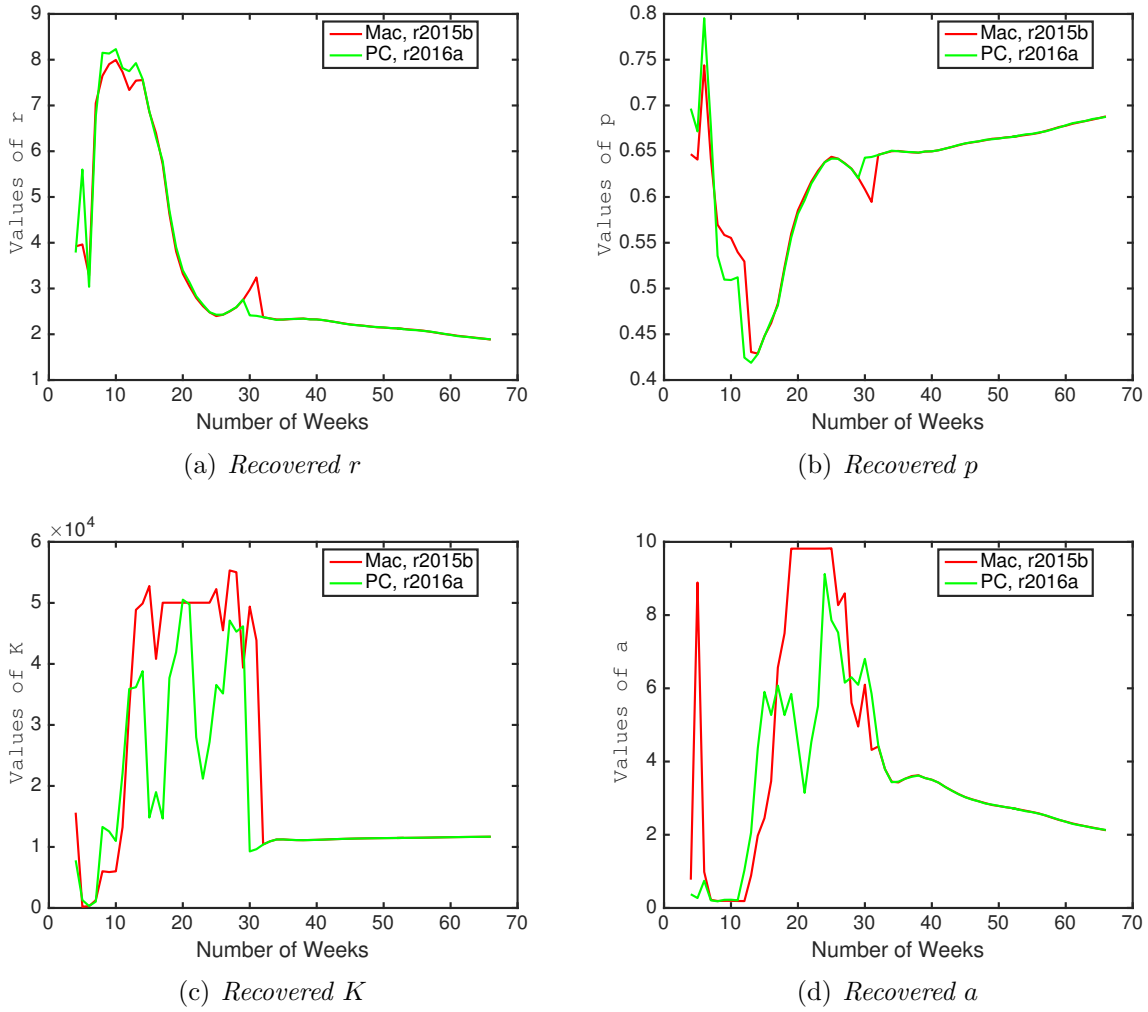


(a) *Recovered* r (b) *Recovered* p

(c) *Recovered* K (d) *Recovered* a

Figure 2.3. *Impact of Coding Differences on Parameter Estimation*

10000, $a = 1$, but the vectors have different order of coordinates: $[r, K, p, a]$ and $[r, a, p, K]$. With the exception of this difference, the two codes are identical and are executed on the same computer system. In both cases, the built-in least-squares curve fitting (lsqcurvefit) Matlab sub-function implements the trust-region optimization procedure. Values along the horizontal axis show the number of weeks, for which cumulative data is available to recover the unknown parameters. The corresponding values of the function represent the computed parameters r, p, K and a. For each partial data set, system parameters are assumed to be constant. We note that p and r are relatively consistent between implementations; however the values of K vary greatly. K is an important parameter when it comes to forecasting the



(a) *Recovered r* (b) *Recovered p*

(c) *Recovered K* (d) *Recovered a*

Figure 2.4. *Impact of Differences in Computer Architectures and Versions of Matlab on Parameter Estimation*

potential damage that can be inflicted by an emerging outbreak; the instability noted in a has less impact as it helps with curve fitting and adjusting to asymmetry.

Computer architecture and the version of Matlab utilized also have an effect on the values of K with this being a reflection of instability of the least squares problem. Figure 2.4 gives parameter outputs under two scenarios: (1) 1.7 GHz Intel Core i5 MacAir under OS 10.11.2 running Matlab r2015b; and (2) PC under Windows 10 and Matlab r2016a. Again we observe minor differences in p and r with more significant variation in K.

The instability seen in the recovery of K, may be partially attributed to its lack of effect on the data in the early stages of an outbreak. As previously mentioned, at the onset of the

disease, (2.5) can be approximated by the simplified differential equation

$$C'(t) = rC^p,$$

since for this time period, with cumulative cases being a very small fraction of the total outbreak capacity, $(C/K)^a$ is small. It is understandable to see a wide range of resulting values for K in the early time period of an outbreak when it is recovered from the data on which its value has little impact.

2.3 A creative formulation

The above experiments indicate that estimation of K is rather unstable. Therefore our next step is to eliminate K from the least squares problem (LSP) [59], and to replace it with another closely related (and equally important) parameter, τ, the disease turning point, which is much closer to other parameters in its order of magnitude. To this end, instead of using the initial condition at t_1 to identify the desired solution curve of (2.5), we take the value of C at τ, and reformulate the initial value problem (2.5) as follows

$$\frac{dC}{dt} = rC^p\left[1 - \left(\frac{C}{K}\right)^a\right] \implies \frac{1}{K}\frac{dC}{dt} = \frac{r}{K^{1-p}}\left(\frac{C}{K}\right)^p\left[1 - \left(\frac{C}{K}\right)^a\right]. \tag{2.7}$$

We define

$$b := \frac{r}{K^{1-p}}, \quad H(t) := \frac{C(t)}{K}$$

to obtain

$$\frac{dH}{dt} = bH^p\left(1 - H^a\right). \tag{2.8}$$

We differentiate (2.8) and set it equal to zero to determine, τ, the inflection point of the modified differential equation.

$$\frac{d^2H}{dt^2} = (bpH^{p-1} - b(a+p)H^{a+p-1})\frac{dH}{dt} = bH^p\frac{dH}{dt}\left(\frac{p - (a+p)H^a}{H}\right).$$

This leads to

$$0 = p - (a+p)H^a \implies H(\tau) = \left(\frac{p}{p+a}\right)^{\frac{1}{a}}, \tag{2.9}$$

which combined with (2.8) gives us the boundary value problem

$$\frac{dH}{dt} = bH^p\left(1 - H^a\right), \quad H(\tau) = \left(\frac{p}{a+p}\right)^{\frac{1}{a}}. \tag{2.10}$$

The BVP is solved at every step of the optimization algorithm on some interval $[t_1, t_m]$ that may or may not contain τ. Numerically, we solve it as an IVP by built-in ode23s in the following sense with two cases to be considered:

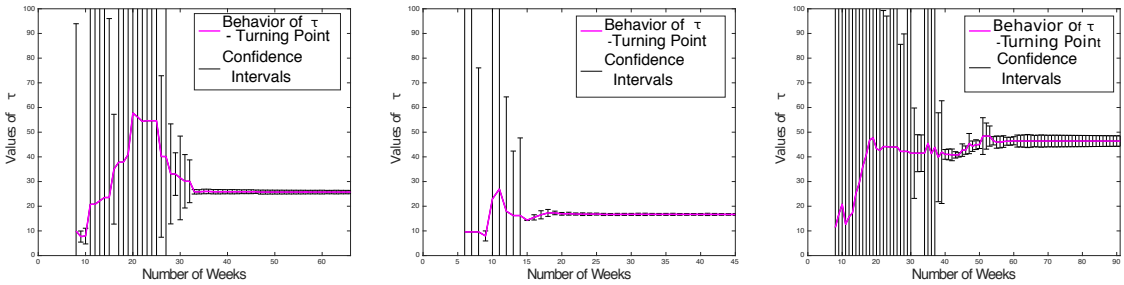
1. If $\tau < t_m$, then we solve the ODE forward on the interval $[\tau, t_m]$ using the initial value condition at τ. Subsequently, we solve the ODE backwards on $[\tau, t_1]$ with a negative step size, again utilizing the initial condition. This yields numerical solutions at the grid points $\{t_1, t_2, ..., t_m\}$.

2. If $\tau \geq t_m$, then we solve the ODE backwards on the interval $[\tau, t_1]$ as before. The extraneous entries from the solution vector (after t_m) are deleted so that we have numerical solutions at $\{t_1, t_2, ..., t_m\}$ only.

To ensure that early cases do not dominate over the later ones (that are usually less noise contaminated), we replace cumulative data for an epidemic with the incidence data $I = [I_1, I_2, ..., I_m]$. By solving (2.10) as stated above, for each set of paramters one obtains numerical values of the derivative, $\frac{dH}{dt}$, at $\{t_1, t_2, ..., t_m\}$. Since these values approximate the corresponding normalized incidence data, we have the following non-linear least squares problem:

$$\min_{b,p,a,K,\tau} \frac{1}{2} \left\| K\frac{dH}{dt}(b, p, a, \tau) - I \right\|^2.$$

Seemingly, we face a more challenging problem as we now have five parameters rather than



(a) *Sierra Leone* (b) *Liberia* (c) *Guinea*

Figure 2.5. *Turning Point Numerical Results utilizing Generalized Richards Model and MATLAB lsqcurvefit for Recovery and MATLAB nlparci for Confidence Intervals*

four. However, K can be eliminated. Indeed, let

$$f := \frac{1}{2}\left\| K\frac{dH}{dt}(b,p,a,\tau) - I \right\|^2 = \frac{1}{2}K^2\left\|\frac{dH}{dt}\right\|^2 - K\left(\frac{dH}{dt}, I\right) + \frac{1}{2}\|I\|^2.$$

By the first order necessary condition,

$$\frac{\partial f}{\partial K} = K\left\|\frac{dH}{dt}\right\|^2 - \left(\frac{dH}{dt}, I\right) = 0,$$

which implies

$$K = \frac{1}{\left\|\frac{dH}{dt}\right\|^2}\left(\frac{dH}{dt}, I\right). \tag{2.11}$$

Thus, we have the LSP with respect to parameters b, p, a, τ only, since $K = K(b,p,a,\tau)$. From now on, we denote $\mathbf{q} := [b,p,a,\tau]^T$ as the parameter vector. Formally, the revised least squares problem is

$$\min_{\mathbf{q}} \frac{1}{2}\left\| K(\mathbf{q})\frac{dH}{dt}(\mathbf{q}) - I \right\|^2. \tag{2.12}$$

Once the LSP is solved, we can compute K by (2.11) and obtain $r = bK^{1-p}$.

Figure 2.5 illustrates the behavior of turning point parameter τ as a function of the number of weeks of incidence data, computed using the revised least squares problem and Matlab built-in lsqcurvefit solver. Black vertical bars represent the confidence intervals (CIs) evaluated with Matlab built-in nlparci sub-function, which employs a method based on asymptotic normal distribution for the parameter estimate to obtain the CIs. The outline

of this algorithm is as follows [60]. Let $\bar{\mathbf{q}}$ be an approximate minimizer of (2.12). Calculate the residual variance as

$$\mathcal{V} = \left\| K(\bar{\mathbf{q}}) \frac{dH}{dt}(\bar{\mathbf{q}}) - I \right\|^2 / df,$$

where df is the residual degree of freedom:

$$df = \text{length}(I) - \text{length}(\mathbf{q}).$$

The residual variance, $\mathcal{V}$, along with a suitable approximation for the Jacobian matrix of the least squares residual, J, yields the estimate for the coefficient variance

$$v = \mathcal{V}(J^* J)^{-1}.$$

At the final step, the coefficient variance, v, is used to find the upper and lower confidence bounds,

$$\bar{\mathbf{q}} \pm \text{tinv}(0.975, df) \sqrt{\text{diag}(v)}, \tag{2.13}$$

respectively. In (2.13), $\text{tinv}(\rho, n)$ is the inverse of *t-cdf* (cumulative distribution function) with its first parameter, ρ, being the desired probability, and the second parameter, n, representing the degree of freedom. In Section 2.5 below, when $\bar{\mathbf{q}}$ is approximated by the Reduced Iteratively Regularized Gauss-Newton scheme, the Jacobian matrix in nlparci is replaced with its reduced version.

Our numerical experiments aimed at the recovery of τ from partial data sets reaffirm that models with fewer parameters (such as the classical Logistic model) have shorter intervals of large uncertainty in the reconstructed values of τ. However, the accuracy of τ, prior to the actual turning point, approximated by p-Logistic ($a = 1$) and the generalized Richards model tends to be higher. The model that gives the best results varies among data sets; for Sierra Leone the p-Logistic model gives the best result, while for Guinea and Liberia the generalized Richards model outperforms.

Very similar results have been obtained with optimization executed by the classical

iteratively regularized Gauss-Newton (IRGN) algorithm [44, 61–66], which provides us with more control over regularization compared to the Matlab lsqcurvefit built-in procedure. Thus while replacing parameter K with τ does improve the efficiency of the numerical scheme, the instability of K is essentially carried into the new parameter τ and, therefore, further analysis of the numerical method is required.

2.4 Motivation for truncating the Jacobian

Due to severe noise propagation in the parameter τ prior to the actual turning point, which is evident from the large confidence intervals, sporadic behavior, and ill-conditioned Jacobians at each step, our next goal is to consider the computational properties of the gradient and Hessian approximation and to design a more problem-oriented regularized procedure to estimate τ at the early stages of an epidemic.

Recall that to approximate τ and other unknown parameters, we consider the constrained least squares problem

$$\min_{\mathbf{q},H} \frac{1}{2} \left\| K(\mathbf{q}) \frac{dH}{dt} - I \right\|^2, \quad \text{subject to } F(\mathbf{q}, H) = \mathbf{0},$$

where the operator F is defined by the ODE and by the boundary value condition at τ. When BVP (2.10) is solved numerically, we define

$$\Phi(\mathbf{q}) := K(\mathbf{q}) \frac{dH}{dt}(\mathbf{q}), \quad \Phi : \mathbb{R}^4 \to \mathbb{R}^m, \tag{2.14}$$

and penalize the cost functional to obtain the unconstrained regularized least squares problem (variational Tikhonov's regularization) [7, 67, 68]

$$\min_{\mathbf{q}} f_\alpha(\mathbf{q}) := \min_{\mathbf{q}} \frac{1}{2} \|\Phi(\mathbf{q}) - I\|^2 + \frac{\alpha}{2} \|L(\mathbf{q} - \tilde{\mathbf{q}})\|^2. \tag{2.15}$$

Here L is a linear operator, $L : \mathbb{R}^4 \to \mathbb{R}^n$, $n \geq 4$ and $\tilde{\mathbf{q}}$ is a reference value of $\mathbf{q}$. By solving (2.15) with the Gauss-Newton algorithm and updating α iteratively, we get the classical

IRGN procedure [61]

$$\left[\Phi'^{*}(\mathbf{q}_k)\Phi'(\mathbf{q}_k) + \alpha_k L^* L\right]\mathbf{p}_k = -\left[\Phi'^{*}(\mathbf{q}_k)\left(\Phi(\mathbf{q}_k) - I\right) + \alpha_k L^* L\left(\mathbf{q}_k - \tilde{\mathbf{q}}\right)\right], \tag{2.16}$$

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \lambda_k \mathbf{p}_k, \quad \lambda_k > 0.$$

In (2.16), α_k is a regularizing sequence that converges to zero as k approaches infinity, $\mathbf{p}_k$ is the direction of the next step and λ_k is a line search parameter. In order to compute the Jacobian $\Phi'(\mathbf{q}_k)$, we evaluate partials of Φ with respect to q_i

$$\frac{\partial \Phi_j}{\partial q_i} = \frac{\partial K}{\partial q_i}(\mathbf{q})\frac{dH(t_j)}{dt} + K(\mathbf{q})\frac{d}{dt}\frac{\partial H(t_j)}{\partial q_i}. \tag{2.17}$$

To find partials of H, we differentiate the ODE in (2.10) with respect to each parameter to form a system of five differential equations to be solved numerically. In this system, the first differential equation is the original ODE with its corresponding boundary condition at τ. The remaining four differential equations are for $\frac{\partial H}{\partial q_i}$

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial H}{\partial b}\right) &= bH^p\left[\frac{1-H^a}{b} + \frac{\partial H}{\partial b}\left(\frac{p}{H} - (a+p)H^{a-1}\right)\right], \\
\frac{d}{dt}\left(\frac{\partial H}{\partial p}\right) &= bH^p\left[\ln(H) + \frac{p}{H}\frac{\partial H}{\partial p} - H^a\left(\ln(H) + \frac{a+p}{H}\frac{\partial H}{\partial p}\right)\right], \\
\frac{d}{dt}\left(\frac{\partial H}{\partial a}\right) &= bH^p\left[\frac{p}{H}\frac{\partial H}{\partial a} - H^a\left(\ln(H) + \frac{a+p}{H}\frac{\partial H}{\partial a}\right)\right], \\
\frac{d}{dt}\left(\frac{\partial H}{\partial \tau}\right) &= bH^p\frac{\partial H}{\partial \tau}\left[\frac{p}{H} - (a+p)H^{a-1}\right],
\end{aligned}$$

where the boundary conditions at τ are obtained by using the boundary condition in (2.10):

$$\begin{aligned}
\frac{\partial H(\tau)}{\partial b} &= 0, \\
\frac{\partial H(\tau)}{\partial p} &= \frac{p^{\frac{1}{a}-1}}{(a+p)^{\frac{1}{a}+1}}, \\
\frac{\partial H(\tau)}{\partial a} &= -\left(\frac{p}{a+p}\right)^{\frac{1}{a}} \frac{(a+p)\ln\left(\frac{p}{a+p}\right)+a}{a^2(a+p)}, \\
\frac{\partial H(\tau)}{\partial \tau} &= -b\left(\frac{p}{a+p}\right)^{\frac{p}{a}} \frac{a}{a+p}.
\end{aligned}$$

Upon obtaining the partials, we can evaluate (2.17) at each point in time t_j to form the Jacobian Φ', which enables us to construct both the gradient and the Hessian approximation. As we transition from full Newton's to the Gauss-Newton method (by dropping the terms involving the second derivatives of $\phi(\mathbf{q})$), the approximate Hessian matrix, G, is calculated as follows

$$\begin{aligned}
G_{ij}(\mathbf{q}) &= \left(\frac{\partial \Phi}{\partial q_i}(\mathbf{q}), \frac{\partial \Phi}{\partial q_j}(\mathbf{q})\right) \\
&= \left(\frac{\partial K}{\partial q_i}(\mathbf{q})\frac{dH}{dt}(\mathbf{q}) + K(\mathbf{q})\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), \frac{\partial K}{\partial q_j}(\mathbf{q})\frac{dH}{dt}(\mathbf{q}) + K(\mathbf{q})\frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q})\right),
\end{aligned}$$

and for $\frac{\partial K}{\partial q_i}(\mathbf{q})$ one has

$$\frac{\partial K}{\partial q_i}(\mathbf{q}) = \frac{\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), I\right) - 2K(\mathbf{q})\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)}{\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)},$$

which, when substituted in the above identity, gives us

$$G_{ij}(\mathbf{q}) = \frac{\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), I\right)\left(\frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q}), I\right)}{\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)} - K(\mathbf{q})\frac{\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), I\right)\left(\frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)}{\left(\frac{dH}{dt}(\mathbf{q})\right), \frac{dH}{dt}(\mathbf{q})\right)}$$
$$- K(\mathbf{q})\frac{\left(\frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q}), I\right)\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)}{\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)} + K^2(\mathbf{q})\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), \frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q})\right). \quad (2.18)$$

To better understand the reasons for instability, we rearrange the terms in (2.18) to obtain

$$G_{ij}(\mathbf{q}) = \frac{1}{\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)}\left\{\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), I\right)\left(\frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q}), I - K(\mathbf{q})\frac{dH}{dt}(\mathbf{q})\right)\right.$$
$$\left. - \left(\frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q}), I\right)\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), K(\mathbf{q})\frac{dH}{dt}(\mathbf{q})\right)\right\} + K^2(\mathbf{q})\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), \frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q})\right).$$

By the Cauchy-Schwarz inequality, the second term inside the braces can be estimated from below as follows

$$-\left(\frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q}), I\right)\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), K(\mathbf{q})\frac{dH}{dt}(\mathbf{q})\right) \geq -K(\mathbf{q})\left(\frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q}), \frac{d}{dt}\frac{\partial H}{\partial q_j}(\mathbf{q})\right)^{1/2}(I, I)^{1/2}$$
$$\cdot \left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), \frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q})\right)^{1/2}\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)^{1/2}.$$

For the diagonal entries of the Hessian approximation, G, this yields

$$G_{ii}(\mathbf{q}) \geq \frac{1}{\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)}\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), I\right)\left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), I - K(\mathbf{q})\frac{dH}{dt}(\mathbf{q})\right)$$
$$- \left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), \frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q})\right)\left[\left\{\frac{\left(I - K(\mathbf{q})\frac{dH}{dt}(\mathbf{q}), I\right) + K(\mathbf{q})\left(\frac{dH}{dt}(\mathbf{q}), I\right)}{\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)}\right\}^{1/2} K(\mathbf{q}) - K^2(\mathbf{q})\right].$$

Taking into consideration (2.11) and (2.14), one concludes

$$G_{ii}(\mathbf{q}) \geq \frac{1}{\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)} \left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), I\right) \left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), I - \Phi(\mathbf{q})\right) - \left(\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), \frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q})\right) \left[\left\{\frac{(I - \Phi(\mathbf{q}), I)}{\left(\frac{dH}{dt}(\mathbf{q}), \frac{dH}{dt}(\mathbf{q})\right)} + K^2(\mathbf{q})\right\}^{1/2} K(\mathbf{q}) - K^2(\mathbf{q})\right].$$

This lower bound plays an important role in our understanding of the ill-posedness of the inverse problem at hand. Indeed, it is clear that, as we iterate, the residual, $\|I - \Phi(\mathbf{q})\|$, is decreasing (provided that the algorithm converges). Therefore the diagonal elements of the Hessian approximation can become close to zero making the process computationally unstable with G being singular to working precision or highly ill-conditioned. We encountered this problem in the course of our numerical simulations with some limited data sets.

2.5 Numerical study of the Reduced Iteratively Regularized Gauss-Newton (RIRGN) algorithm

Evidently, we face a dilemma of either using the above Hessian approximation coupled with a large penalty term, which reduces the accuracy of the method, or modifying the Hessian to make the algorithm more stable, yet accurate. First we evaluate the gradient of f, $\nabla f(\mathbf{q}) = \Phi'(\mathbf{q})^*(\Phi(\mathbf{q}) - I)$, with the ith component being

$$\frac{\partial f}{\partial q_i}(\mathbf{q}) = \left(\frac{\partial \Phi}{\partial q_i}(\mathbf{q}), \Phi(\mathbf{q}) - I\right) = \left(\frac{\partial K}{\partial q_i}(\mathbf{q})\frac{dH}{dt}(\mathbf{q}) + K(\mathbf{q})\frac{d}{dt}\frac{\partial H}{\partial q_i}(\mathbf{q}), K(\mathbf{q})\frac{dH}{dt}(\mathbf{q}) - I\right).$$

The Jacobian $\Phi'(\mathbf{q})$ can be expressed as the sum of two matrices $\Phi'_1(\mathbf{q}) + \Phi'_2(\mathbf{q})$ in the following manner

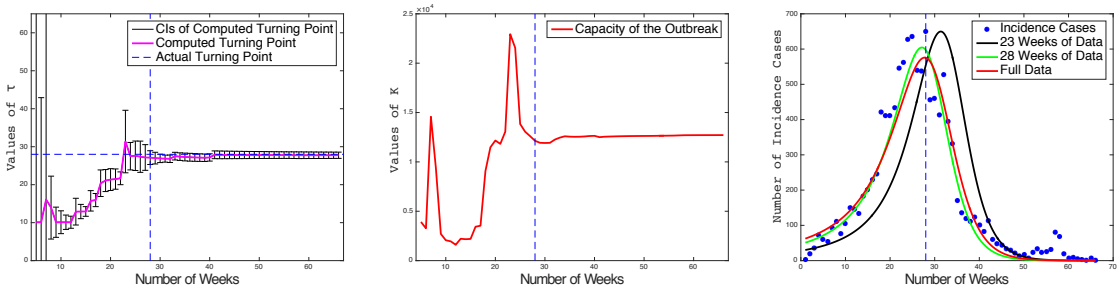
$$\Phi'(\mathbf{q}) = \left[\begin{array}{ccc} \frac{\partial K}{\partial q_1}\frac{dH}{dt} & \cdots & \frac{\partial K}{\partial q_4}\frac{dH}{dt} \end{array}\right] + K\left[\begin{array}{ccc} \frac{d}{dt}\frac{\partial H}{\partial q_1} & \cdots & \frac{d}{dt}\frac{\partial H}{\partial q_4} \end{array}\right] := \Phi'_1(\mathbf{q}) + \Phi'_2(\mathbf{q}). \qquad (2.19)$$

Note that for each i, one has

$$\left(\frac{\partial K}{\partial q_i}(\mathbf{q})\frac{dH}{dt}(\mathbf{q}), K(\mathbf{q})\frac{dH}{dt}(\mathbf{q}) - I\right) = \frac{\partial K}{\partial q_i} K \left\|\frac{dH}{dt}\right\|^2 - \frac{\partial K}{\partial q_i}\left(\frac{dH}{dt}, I\right)$$
$$= \frac{\partial K}{\partial q_i}\left[\frac{1}{\left\|\frac{dH}{dt}\right\|^2}\left(\frac{dH}{dt}, I\right)\left\|\frac{dH}{dt}\right\|^2 - \left(\frac{dH}{dt}, I\right)\right] = 0.$$

Hence, the residual, $\Phi(\mathbf{q}) - I$, is in the kernel of matrix $\Phi_1'(\mathbf{q})$, which yields a simplified form of the gradient, $\nabla f(\mathbf{q}) = \Phi_2'^*(\mathbf{q})(\Phi(\mathbf{q}) - I)$ with a reduced number of operations and, therefore, a reduced noise propagation due to unnecessary rounding. Moreover, the above observation implies that the Hessian approximation comes down to

$$\Phi_2'^*(\mathbf{q})\Phi'(\mathbf{q}) = \Phi_2'^*(\mathbf{q})(\Phi_1'(\mathbf{q}) + \Phi_2'(\mathbf{q})). \tag{2.20}$$

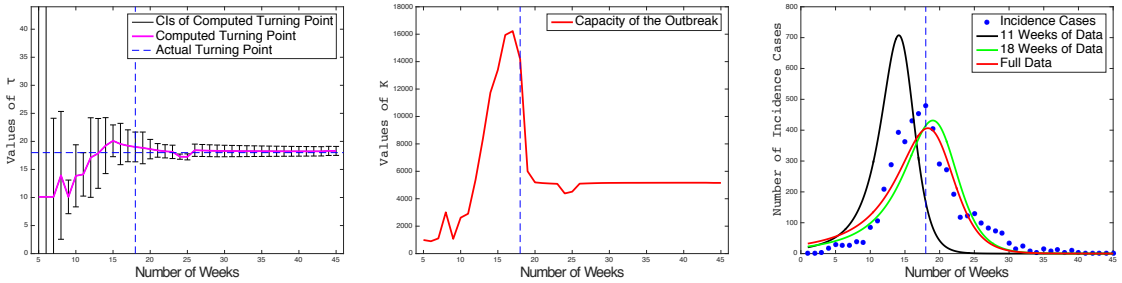


(a) *Estimation of* τ (b) *Computed values of* K (c) *Reconstructed Curves*

Figure 2.6. *Numerical Results for Sierra Leone - Reduced IRGN*

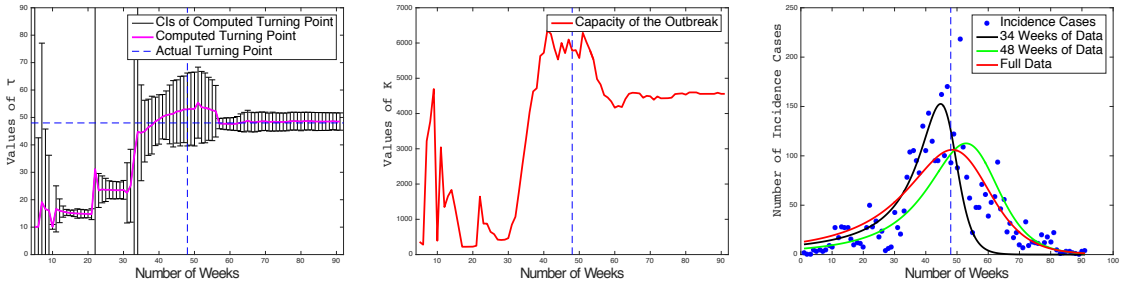
However, $\Phi_2'^*(\Phi_1' + \Phi_2')$ inherits poor computational properties from $\Phi'^*\Phi'$, as one can easily verify by deriving a similar lower bound for operator (2.20) using the Cauchy-Schwarz inequality. Besides, (2.20) is no longer symmetric non-negative definite. This consideration coupled with the evidence from our numerical experiments suggest further reduction of the Hessian approximation by eliminating Φ_1' from (2.20) and setting

$$G(\mathbf{q}) \approx \Phi_2'^*(\mathbf{q})\Phi_2'(\mathbf{q}). \tag{2.21}$$



(a) *Estimation of* τ (b) *Computed values of* K (c) *Reconstructed Curves*

Figure 2.7. *Numerical Results for Liberia - Reduced IRGN*



(a) *Estimation of* τ (b) *Computed values of* K (c) *Reconstructed Curves*

Figure 2.8. *Numerical Results for Guinea - Reduced IRGN*

This operator is symmetric and non-negative definite. Approximation (2.21) results in the following iterative scheme

$$\left[\Phi_2'^*(\mathbf{q}_k)\Phi_2'(\mathbf{q}_k) + \alpha_k L^* L\right]\mathbf{p}_k = -\left[\Phi_2'^*(\mathbf{q}_k)\left(\Phi(\mathbf{q}_k) - I\right) + \alpha_k L^* L(\mathbf{q}_k - \tilde{\mathbf{q}})\right], \tag{2.22}$$

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \lambda_k \mathbf{p}_k, \quad \lambda_k > 0. \tag{2.23}$$

To optimize the step size in (2.22), we use a version of the Armijo-Goldstein line search strategy [69], i.e., a backtracking with $\lambda_k = 1/2, 1/4, ...$ until

$$\|\Phi(\mathbf{q}_k + \lambda_k \mathbf{p}_k) - I\|^2 < \|\Phi(\mathbf{q}_k) - I\|^2 + \lambda_k \beta(\Phi_2'^*(\mathbf{q}_k)(\Phi(\mathbf{q}_k) - I), \mathbf{p}_k),$$

which is commonly implemented for Gauss-Newton type algorithms. In (2.22), we assume that L^*L is invertible and

$$(L^* L\mathbf{h}, \mathbf{h}) \geq c||\mathbf{h}||^2, \quad c > 0, \tag{2.24}$$

for any $\mathbf{h} \in \mathbb{R}^4$. The upgrade from the identity operator to a general linear operator L allows, when necessary, the placement of more regularization on some unknown parameters and less on others. Condition (2.24) includes a bound which depends on the finite dimensional operator L in terms of calculable parameter $c = \min \varsigma_i^2$, which in turn enters in (2.32), used for the convergence analysis in the next section, through its inverse. Here ς_i are the singular values of L ordered from largest to smallest. For the version of L suggested below, $c = 1$.

We call (2.22) *Reduced Iteratively Regularized Gauss-Newton*. For our specific problem, this algorithm is more stable compared to classical IRGN, and most solution curves obtained by (2.22) are superior to those produced by IRGN or Matlab built-in lsqcurvefit in terms of accuracy and stability as one can see by comparing reconstructions of τ in Figure 2.5 and Figures 2.6-2.8.

In particular, Figures 2.6-2.8 illustrate the advantages of the Reduced IRGN in leading to much smaller confidence intervals (CIs) in the decisive time period before the turning point. The new method does lead slightly larger CIs after the turning point, but this period is much less important in practice.

Apart from the turning points, τ, the initial results for Sierra Leone, Guinea, and Liberia presented in Figures 2.6-2.8 illustrate saturation levels, K, and comparison of the forecasting curves with parameters recovered at (a) the earliest possible moment (black curves), (b) at the actual turning point (green curves), and (c) from full disease data (red curves). For the parameter K, RIRGN yields considerably more accurate upper bounds prior to the turning points as compared to our initial reconstructions shown in Figure 2.5. All experiments presented in Figures 2.6-2.8 have been conducted for the generalized Richards model.

As we implement algorithm (2.22) in practice, at every step of the iterative process $K^2(\mathbf{q}_k)$ is canceled on both sides of (2.22), which yields the following system

$$\left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]\mathbf{p}_k = -\left[A'^*(\mathbf{q}_k)\left(A(\mathbf{q}_k) - I_\delta/K(\mathbf{q}_k)\right) + \tilde{\alpha}_k L^*L(\mathbf{q}_k - \tilde{\mathbf{q}})\right], \quad (2.25)$$

where $A(\mathbf{q}) := \frac{dH}{dt}(\mathbf{q})$, $||I - I_\delta|| \leq \delta$, and $\tilde{\alpha}_k := \alpha_k/K^2(\mathbf{q}_k)$. Recall that $K(\mathbf{q}_k)$ in (2.11)

is evaluated from noisy data. Hence division by $K^2(\mathbf{q}_k)$ enables us to move all noise from the matrix of system (2.25) to its right-hand side, which makes (2.25) computationally more stable. It also normalizes the residual and allows the use of the same regularization sequence, $\{\tilde{\alpha}_k\}$, for multiple data sets. The only adjustment that needs to be made is for $\tilde{\alpha}_0$, since data sets with higher noise level require more regularization. In all experiments shown in Figures 2.6-2.8, $\tilde{\alpha}_k = \tilde{\alpha}_0 \exp(-k/2)$ with $\tilde{\alpha}_0 = 5 \cdot 10^{-4}$ for Sierra Leone and Liberia, and $\tilde{\alpha}_0 = 10^{-3}$ for Guinea. The choice $\tilde{\alpha}_k = \tilde{\alpha}_0 \exp(-k/2)$ provides the most aggressive convergence rate for the regularized algorithm. At the same time, it preserves stability at every step of the iterative process until it is terminated by stopping rule (2.28) below. The stopping rule guarantees that, while our numerical solution does fit the data, we do not over-fit and ensure approximation of the exact solution to the noise-free problem rather than solution to the problem with noise-contaminated data. This phenomena is called semi-convergence, and stopping at the right moment is crucial for an unstable model. Stopping rule (2.28) is explained in the next section.

Another important aspect is the choice of L in (2.25). In the vector $\mathbf{q} := [b, p, a, \tau]^T$, the value of τ is between one and two orders of magnitude larger than b, p, and a. This suggests that the regularization applied to b, p, and a should be appropriately weighted in order to balance the sensitivity of the cost functional to all four parameters. Thus we take

$$L = \begin{bmatrix} \omega & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & \omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \omega > 1. \tag{2.26}$$

An arbitrary choice, say, $\omega = 10$ gives stable computational results, but $\omega = 1$ yields a very poor accuracy of the approximate solutions, since either τ tends to be over-regularized or there is lack of stability in b, p, and a. For this reason, the use of a general linear operator L, rather than the identity operator, is crucial for the success of the proposed algorithm.

The choice of the test function, $\tilde{\mathbf{q}}$, that is meant to bring *a priori* information in the

penalty term, is very difficult. In the beginning of an emerging outbreak, it is hard to have an accurate *a priori* estimate as to when the peak is going to occur. For a fair comparison to lsqcurvefit, in all our experiments we take $\tilde{\mathbf{q}} = [1, 1, 1.5, 60]$, i.e., we assume the incidence curve will turn after 60 weeks, which puts Liberia at huge disadvantage, since the actual turning point there appears to be 18. However, the use of a general matrix L in the penalty term allows us to reduce the weight on τ and, nevertheless, maintain stability. As the result, the poor *a priori* value of $\tilde{\mathbf{q}}$ does not hinder the recovery of τ in case of Liberia. In fact, the reconstruction of the Liberia turning point is the most accurate, partly due to a smaller noise level (compared to, say, Guinea) and partly because of a shorter time frame. The most difficult case is Guinea due to a high noise level in the reported incidence data. But even for Guinea, the τ curve does not bounce all the way towards 60 (as opposed to lsqcurvefit where 60 is enforced as upper bound on τ) and, starting with week 34, we get a reliable estimate of the actual turning point, 48.

For all numerical experiments presented in this paper, the confidence intervals have been computed with Matlab built-in nlparci sub-function that estimates uncertainty in the recovered parameters using residual and Hessian approximation for a Newton-type iterative method at hand. The iterations are terminated by the generalized discrepancy principle as outlined in the convergence analysis below. Encouraged by the numerical simulations presented in this section, we move to the theoretical study of the RIRGN procedure.

2.6 Convergence analysis of the RIRGN method

In order to show that the RIRGN algorithm is well-defined and convergent, we use the general scheme developed for the analysis of the original IRGN [44, 61, 63, 70]. Assume that $\{\tilde{\alpha}_k\}$ in (2.25) is a regularization sequence satisfying the conditions

$$\tilde{\alpha}_k \geq \tilde{\alpha}_{k+1} > 0, \qquad \sup_{k \in \mathcal{N} \bigcup \{0\}} \frac{\tilde{\alpha}_k}{\tilde{\alpha}_{k+1}} = d < \infty, \qquad \lim_{k \to \infty} \tilde{\alpha}_k = 0,$$

and $\{\lambda_k\}$ is a step size sequence such that

$$0 < \lambda \leq \lambda_k \leq 1.$$

Let $\hat{\mathbf{q}}$ be a solution to the equation $K(\mathbf{q})\mathbf{A}(\mathbf{q}) - \mathbf{I} = \mathbf{0}$ (maybe non-unique) and let I be given by its noise-contaminated approximation I_δ such that

$$||I - I_\delta|| \leq \delta, \quad \delta \geq 0.$$

It has been established in [61, 63, 70] that, if the following estimate holds

$$||\mathbf{q}_{k+1} - \hat{\mathbf{q}}|| \leq (1 - \gamma\lambda_k)||\mathbf{q}_k - \hat{\mathbf{q}}|| + \frac{\lambda_k\beta}{\sqrt{\tilde{\alpha}_k}}||\mathbf{q}_k - \hat{\mathbf{q}}||^2 + \lambda_k\sqrt{\tilde{\alpha}_k}\sigma + \frac{\lambda_k\kappa\delta}{\sqrt{\tilde{\alpha}_k}}, \quad k = 0, 1, ..., \quad (2.27)$$

for $\{\mathbf{q}_k\}$ in a Hilbert space $\mathbb{H}$ with some non-negative constants β, γ, σ, and κ (with σ being sufficiently small and $\gamma\lambda < 1$), then there exists $l > 0$, $l = l(\beta, \gamma, \sigma, \kappa, d)$ such that

$$\frac{||\mathbf{q}_k - \hat{\mathbf{q}}||}{\sqrt{\tilde{\alpha}_k}} \leq l, \quad \text{for} \quad k = 0, 1, ..., \mathcal{K}(\delta),$$

provided $||\mathbf{q}_0 - \hat{\mathbf{q}}||$ is sufficiently small, and $\mathcal{K} = \mathcal{K}(\delta)$ is evaluated by the discrepancy-type stopping rule

$$||A(\mathbf{q}_{\mathcal{K}(\delta)}) - I_\delta/K(\mathbf{q}_{\mathcal{K}(\delta)})||^2 \leq \rho\kappa\delta < ||A(\mathbf{q}_k) - I_\delta/K(\mathbf{q}_k)||^2, \quad 0 \leq k \leq \mathcal{K}(\delta), \quad \rho > 1, \quad (2.28)$$

and the sequence $\mathcal{K} = \mathcal{K}(\delta)$ is admissible, that is,

$$\lim_{\delta \to 0} ||\mathbf{q}_{\mathcal{K}(\delta)} - \bar{\mathbf{q}}|| = 0,$$

for $\bar{\mathbf{q}} = \text{argmin}_{\mathbf{q} \in \mathbb{H}}||A(\mathbf{q}) - I/K(\mathbf{q})||$.

Remark 2.6.1 *A stronger version of the above stopping rule has been proposed for IRGN in [71] under the assumption that L is the identity operator.*

In what follows, we will verify that for $\{\mathbf{q}_k\}$, defined in (2.25), inequality (2.27) holds. Assume as before that $A(\mathbf{q}) := \frac{dH}{dt}(\mathbf{q})$, and let $A : \mathbb{R}^4 \to \mathbb{R}^m$, where m is the number of data points. Clearly, the matrix $A'(\mathbf{q})$ is Lipschitz continuous in a neighborhood $\mathcal{O}(\hat{\mathbf{q}})$, which does not contain negative values of b, p, a, and τ. Negative values of b, p, a, and τ are not relevant for our particular application. Assume that for any $\mathbf{u}, \mathbf{v} \in \mathcal{O}(\hat{\mathbf{q}})$,

$$||A'(\mathbf{u})|| \leq M_1, \quad ||A'(\mathbf{u}) - A'(\mathbf{v})|| \leq M_2||\mathbf{u} - \mathbf{v}||, \quad \text{and} \quad \frac{||A(\mathbf{u})||}{|(A(\mathbf{u}), A(\mathbf{v}))|} \leq N. \tag{2.29}$$

The last assumption in (2.29) underscores that, while $A(\mathbf{u}) > 0$, $\mathbf{u} \in \mathcal{O}(\hat{\mathbf{q}})$, may get close to zero as C approaches K, we do not consider the case when $t \to \infty$ or gets too large. The time of an outbreak is finite, and $1 - C(t)/K \geq 1 - C(t_m)/K > 0$. For early data, t_m is even smaller than in the case when the entire outbreak is investigated.

To deduce a bound for $A(\hat{\mathbf{q}}) - A(\mathbf{q}_k) - A'(\mathbf{q}_k)(\hat{\mathbf{q}} - \mathbf{q}_k)$, we use the second assumption in (2.29). Let

$$\phi(t) := A(\mathbf{v} + t(\mathbf{u} - \mathbf{v})), \quad \text{then} \quad \phi(0) = A(\mathbf{v}), \quad \text{and} \quad \phi(1) = A(\mathbf{u})$$

so that

$$\phi(1) - \phi(0) = \int_0^1 \phi'(t)\, dt \implies A(\mathbf{u}) - A(\mathbf{v}) = \int_0^1 A'(\mathbf{v} + t(\mathbf{u} - \mathbf{v}))(\mathbf{u} - \mathbf{v})\, dt.$$

Thus one arrives at the following estimates

$$\begin{aligned}
||A(\mathbf{u}) - A(\mathbf{v}) - A'(\mathbf{v})(\mathbf{u}-\mathbf{v})|| &= \left\| \int_0^1 A'(\mathbf{v}+t(\mathbf{u}-\mathbf{v}))(\mathbf{u}-\mathbf{v})\,dt - A'(\mathbf{v})(\mathbf{u}-\mathbf{v}) \right\| \\
&= \left\| \int_0^1 \{A'(\mathbf{v}+t(\mathbf{u}-\mathbf{v})) - A'(\mathbf{u})\}\,dt(\mathbf{u}-\mathbf{v}) \right\| \\
&\leq \int_0^1 \|A'(\mathbf{v}+t(\mathbf{u}-\mathbf{v})) - A'(\mathbf{u})\|\,dt\,\|(\mathbf{u}-\mathbf{v})\| \\
&\leq M_2 \int_0^1 \|\mathbf{v}+t(\mathbf{u}-\mathbf{v})) - \mathbf{v}\|\,dt\,\|(\mathbf{u}-\mathbf{v})\| \\
&= M_2 \int_0^1 t\,dt\,\|(\mathbf{u}-\mathbf{v})\|^2 = \frac{M_2}{2}\,\|(\mathbf{u}-\mathbf{v})\|^2 .
\end{aligned}$$

This yields

$$A(\hat{\mathbf{q}}) = A(\mathbf{q}_k) + A'(\mathbf{q}_k)(\hat{\mathbf{q}} - \mathbf{q}_k) + \mathcal{B}(\hat{\mathbf{q}}, \mathbf{q}_k), \tag{2.30}$$

where

$$||\mathcal{B}(\hat{\mathbf{q}}, \mathbf{q}_k)|| \leq \frac{M_2}{2}||\hat{\mathbf{q}} - \mathbf{q}_k||^2.$$

By (2.25) and (2.29), one concludes that

$$\begin{aligned}
\mathbf{q}_{k+1} - \hat{\mathbf{q}} = \mathbf{q}_k - \hat{\mathbf{q}} &- \lambda_k \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} \left\{A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right\}(\mathbf{q}_k - \hat{\mathbf{q}}) \\
&- \lambda_k \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} A'^*(\mathbf{q}_k)\left\{A(\hat{\mathbf{q}}) - I_\delta/K(\mathbf{q}_k) - \mathcal{B}(\hat{\mathbf{q}}, \mathbf{q}_k)\right\} \\
&- \lambda_k\tilde{\alpha}_k \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} L^*L(\hat{\mathbf{q}} - \tilde{\mathbf{q}}).
\end{aligned} \tag{2.31}$$

As proven in [61, 63, 70], under the assumption (2.24)

$$\left\| \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} \right\| \leq \frac{1}{\tilde{\alpha}_k c}, \tag{2.32}$$

$$\left\| \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} A'^*(\mathbf{q}_k) \right\| \leq \frac{1}{2\sqrt{\tilde{\alpha}_k c}}. \tag{2.33}$$

To estimate $\left\| \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) \right\|$, we proceed as follows. Let $D :=$

$A'^*(\mathbf{q}_k)A'(\mathbf{q}_k)$, $B := L^*L$ and $C := D^{1/2}B^{-1/2}$ then

$$\begin{aligned}
||\left[D + \tilde{\alpha}_k B\right]^{-1} D|| &= ||\left\{B^{1/2}\left[B^{-1/2}DB^{-1/2} + \tilde{\alpha}_k I\right]B^{1/2}\right\}^{-1} D|| \\
&= ||B^{-1/2}\left[B^{-1/2}D^{1/2}D^{1/2}B^{-1/2} + \tilde{\alpha}_k I\right]^{-1} B^{-1/2}D^{1/2}D^{1/2}B^{-1/2}B^{1/2}|| \\
&= ||B^{-1/2}\left[C^*C + \tilde{\alpha}_k I\right]^{-1} C^*CB^{1/2}|| \\
&\leq \|B^{-1/2}\| \, \|\left[C^*C + \tilde{\alpha}_k I\right]^{-1} C^*C\| \, \|B^{1/2}\| \leq \mathrm{cond}(B^{1/2}) =: \zeta,
\end{aligned}$$

which implies that

$$\left\|[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L]^{-1} A'^*(\mathbf{q}_k)A'(\mathbf{q}_k)\right\| \leq \zeta. \tag{2.34}$$

Consider the term $A(\hat{\mathbf{q}}) - I_\delta/K(\mathbf{q}_k)$ in (2.31). One has

$$A(\hat{\mathbf{q}}) - I_\delta/K(\mathbf{q}_k) = A(\hat{\mathbf{q}}) - I/K(\mathbf{q}_k) + (I - I_\delta)/K(\mathbf{q}_k). \tag{2.35}$$

Since $K(\hat{\mathbf{q}})A(\hat{\mathbf{q}}) - I = 0$, one writes $K(\mathbf{q}_k)$ as

$$K(\mathbf{q}_k) = \frac{(A(\mathbf{q}_k), I)}{(A(\mathbf{q}_k), A(\mathbf{q}_k))} = K(\hat{\mathbf{q}})\frac{(A(\mathbf{q}_k), A(\hat{\mathbf{q}}))}{(A(\mathbf{q}_k), A(\mathbf{q}_k))}. \tag{2.36}$$

Based on (2.36), one derives

$$A(\hat{\mathbf{q}}) - I/K(\mathbf{q}_k) = A(\hat{\mathbf{q}}) - \frac{I(A(\mathbf{q}_k), A(\mathbf{q}_k))}{K(\hat{\mathbf{q}})(A(\mathbf{q}_k), A(\hat{\mathbf{q}}))} = A(\hat{\mathbf{q}}) - A(\hat{\mathbf{q}})\frac{(A(\mathbf{q}_k), A(\mathbf{q}_k))}{(A(\mathbf{q}_k), A(\hat{\mathbf{q}}))}. \tag{2.37}$$

If one replaces $\hat{b}$ with zero in $\hat{\mathbf{q}} := [\hat{b}, \hat{p}, \hat{a}, \hat{\tau}]^T$ and introduces the vector $\mathbf{q}^{(b)} := [0, \hat{p}, \hat{a}, \hat{\tau}]^T$, then by the equation in (2.10)

$$0 = A(\mathbf{q}^{(b)}) = A(\hat{\mathbf{q}}) + A'(\xi^{(b)})(\mathbf{q}^{(b)} - \hat{\mathbf{q}}).$$

Suppose that $A'(\xi^{(b)})$ takes the form $A'(\xi^{(b)}) = A'(\hat{\mathbf{q}})R(\xi^{(b)}, \hat{\mathbf{q}})$, with $R(\xi^{(b)}, \hat{\mathbf{q}})$ being a 4×4 matrix and $||R(\xi^{(b)}, \hat{\mathbf{q}})(\mathbf{q}^{(b)} - \hat{\mathbf{q}})|| := \mu$, a small positive constant. This assumption is justified,

since $||\mathbf{q}^{(b)} - \hat{\mathbf{q}}|| = |\hat{b}| = \hat{b}$, and parameter b is normalized by $K^{1-p},\ 1-p > 0$. Therefore for all data sets we consider, $0 < \hat{b} < 1$ (usually it is between 0.2 and 0.3) and in all experiments we take $b_0 = 0.5$. Hence from (2.37) one concludes

$$A(\hat{\mathbf{q}}) - I_\delta/K(\mathbf{q}_k) = A'(\xi^{(b)})(\hat{\mathbf{q}} - \mathbf{q}^{(b)})\frac{(A(\mathbf{q}_k), A(\hat{\mathbf{q}}) - A(\mathbf{q}_k))}{(A(\mathbf{q}_k), A(\hat{\mathbf{q}}))} + (I - I_\delta)/K(\mathbf{q}_k)$$

$$= A'(\hat{\mathbf{q}})R(\xi^{(b)}, \hat{\mathbf{q}})(\hat{\mathbf{q}} - \mathbf{q}^{(b)})\frac{(A(\mathbf{q}_k), A'(\xi_k)(\hat{\mathbf{q}} - \mathbf{q}_k))}{(A(\mathbf{q}_k), A(\hat{\mathbf{q}}))} + (I - I_\delta)/K(\mathbf{q}_k). \tag{2.38}$$

Representation (2.38) implies

$$|| \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} A'^*(\mathbf{q}_k) \left\{A(\hat{\mathbf{q}}) - I_\delta/K(\mathbf{q}_k)\right\} ||$$

$$\leq \Big\{|| \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} A'^*(\mathbf{q}_k)|| \, ||A'(\hat{\mathbf{q}}) - A'(\mathbf{q}_k)|| + || \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k)$$

$$+ \tilde{\alpha}_k L^*L\right]^{-1} A'^*(\mathbf{q}_k)A'(\mathbf{q}_k)|| \Big\} ||R(\xi^{(b)}, \hat{\mathbf{q}})(\hat{\mathbf{q}} - \mathbf{q}^{(b)})|| \, \frac{||A(\mathbf{q}_k)|| \, ||A'(\xi_k)|| \, ||\hat{\mathbf{q}} - \mathbf{q}_k||}{|(A(\mathbf{q}_k), A(\hat{\mathbf{q}}))|}$$

$$+ || \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} A'^*(\mathbf{q}_k)|| \, ||I - I_\delta||/|K(\mathbf{q}_k)|. \tag{2.39}$$

Given the nature of $K(\mathbf{q}_k)$, we assume that $0 < \tilde{K} \leq K(\mathbf{q}_k)$ for any $k = 0, 1, 2, ...$ Inequality (2.39) combined with (2.29) and (2.32) yields

$$|| \left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} A'^*(\mathbf{q}_k) \left\{A(\hat{\mathbf{q}}) - I_\delta/K(\mathbf{q}_k) - \mathcal{B}(\hat{\mathbf{q}}, \mathbf{q}_k)\right\} ||$$

$$\leq \left\{\frac{M_2||\hat{\mathbf{q}} - \mathbf{q}_k||}{2\sqrt{\tilde{\alpha}_k c}} + \zeta\right\} \mu N M_1 ||\hat{\mathbf{q}} - \mathbf{q}_k|| + \frac{\delta}{2\tilde{K}\sqrt{\tilde{\alpha}_k c}} + \frac{M_2||\hat{\mathbf{q}} - \mathbf{q}_k||^2}{4\sqrt{\tilde{\alpha}_k c}}$$

$$= \left[2\mu N M_1 + 1\right] \frac{M_2||\hat{\mathbf{q}} - \mathbf{q}_k||^2}{4\sqrt{\tilde{\alpha}_k c}} + \mu N M_1 \zeta ||\hat{\mathbf{q}} - \mathbf{q}_k|| + \frac{\delta}{2\tilde{K}\sqrt{\tilde{\alpha}_k c}}. \tag{2.40}$$

To complete the estimate for $||\mathbf{q}_{k+1} - \hat{\mathbf{q}}||$, we assume that L and $\tilde{\mathbf{q}}$ are chosen according to the modified source condition [70]

$$L^*L(\hat{\mathbf{q}} - \tilde{\mathbf{q}}) \in A'^*(\hat{\mathbf{q}})S, \quad S := \{w \in \mathbb{R}^m,\ ||w|| \leq \varepsilon\}, \tag{2.41}$$

where ε is a small non-negative constant. If L is the identity operator, this is equivalent to the Hölder-type condition with exponent being $1/2$ [61, 63].

Remark 2.6.2 *To see if assumption (2.41) is reasonable in our case, note that identity (2.25) implies*

$$L^*L(\mathbf{q}_{k+1} - \tilde{\mathbf{q}}) = -A'^*(\mathbf{q}_k)\frac{[A(\mathbf{q}_k) - I_\delta/K(\mathbf{q}_k) - A'(\mathbf{q}_k)(\mathbf{q}_{k+1} - \mathbf{q}_k)]}{\tilde{\alpha}_k} := A'^*(\mathbf{q}_k)w_k. \quad (2.42)$$

Hence, in terms of structure, it is only natural to require that L and $\tilde{\mathbf{q}}$ satisfy (2.41). With our particular choice of L according to (2.26), condition (2.41) does not restrict the unknown parameters to any subspace. On the contrary, it enforces appropriate scaling, which results in a more effective regularization. The appearance of $\tilde{\alpha}_k$ in the denominator of (2.42) highlights the importance of driving $\tilde{\alpha}_k$ to zero at a rate that is not too fast to ensure that w_k remains bounded (accuracy and stability are well balanced).

By (2.41), there is $w \in S$ such that

$$L^*L(\hat{\mathbf{q}} - \bar{\mathbf{q}}) = (A'(\hat{\mathbf{q}}) - A'(\mathbf{q}_k))^*w + A'^*(\mathbf{q}_k)w.$$

This yields the following inequality

$$\tilde{\alpha}_k||\left[A'^*(\mathbf{q}_k)A'(\mathbf{q}_k) + \tilde{\alpha}_k L^*L\right]^{-1} L^*L(\hat{\mathbf{q}} - \tilde{\mathbf{q}})|| \leq \frac{M_2\varepsilon}{c}||\mathbf{q}_k - \hat{\mathbf{q}}|| + \frac{\varepsilon}{2}\sqrt{\frac{\tilde{\alpha}_k}{c}}. \quad (2.43)$$

Taking into account (2.31)-(2.43), one arrives at the estimate

$$||\mathbf{q}_{k+1} - \hat{\mathbf{q}}|| \leq (1 - \lambda_k)||\mathbf{q}_k - \hat{\mathbf{q}}|| + \lambda_k\left[2\mu N M_1 + 1\right]\frac{M_2||\hat{\mathbf{q}} - \mathbf{q}_k||^2}{4\sqrt{\tilde{\alpha}_k c}} + \lambda_k\mu\zeta N M_1||\hat{\mathbf{q}} - \mathbf{q}_k||$$
$$+ \frac{\lambda_k\delta}{2\tilde{K}\sqrt{\tilde{\alpha}_k c}} + \frac{\lambda_k M_2\varepsilon}{c}||\mathbf{q}_k - \hat{\mathbf{q}}|| + \frac{\lambda_k\varepsilon}{2}\sqrt{\frac{\tilde{\alpha}_k}{c}}.$$

Introducing the notations

$$\gamma := 1 - \mu\zeta N M_1 - \frac{M_2\varepsilon}{c}, \quad \kappa := \frac{1}{2\tilde{K}\sqrt{c}}, \quad \beta := \frac{M_2}{4\sqrt{c}}\left[2\mu N M_1 + 1\right], \quad \text{and} \quad \sigma := \frac{\varepsilon}{2\sqrt{c}},$$

we obtain (2.27), which shows convergence of the RIRGN algorithm. □

2.7 Concluding remarks

Stable parameter estimation is an inherently challenging problem in infectious disease modeling, especially from early data. At the onset of an epidemic, quantification of key parameters can help understand the epidemiology of invading pathogen, make predictions of the likely morbidity and mortality impact, as well as disease transmissibility and incidence over time, which in turn could guide a timely implementation of the most effective intervention strategies. For example, as evident from phenomenological models studied here, there is strong correlation between the final size of an epidemic and its turning point, a critical parameter for disease forecasting during the early epidemic growth phase. These models describe the epidemic dynamics in two phases of fast and slow infection spread with a transition (turning) point, at which the maximum rate of disease incidence occurs. In the slow phase of infection spread (after the turning point), the epidemic peaks and subsequently declines, and therefore the cumulative number of cases eventually saturates at the epidemic final size. However, the challenge in parameter estimation generally arises in the fast phase of epidemic spread before the turning point where the amount of data is inadequate given the number of unknown parameters.

In this chapter, our goal was to explore the nature of instability of classical regularized Gauss-Newton-type algorithms for the estimation of important disease parameters at the fast phase of epidemic spread. To enhance computational properties of the Hessian approximation, we introduced a modified problem-oriented optimization procedure, which yields a substantial progress in the recovery of two crucial epidemiological parameters, namely, the epidemic size of an emerging disease and the expected turning point of the outbreak.

The convergence analysis of the new method is proposed under sufficient conditions that are fully justified for the generalized Richards model used to recover these and other unknown parameters.

PART 3

ON STABLE RECONSTRUCTION OF TIME-DEPENDENT TRANSMISSION RATES IN COMPARTMENTAL EPIDEMIC MODELS AND IMPLICATIONS FOR FORECASTING

3.1 Introduction

Real-time reconstruction of disease transmission rates for emerging outbreaks provides crucial information to government agencies working to design and implement public health intervention measures and policies. Despite tremendous progress in both deterministic and stochastic algorithms for solving parameter estimation problems in epidemiology, there is still a long way to go before our understanding of disease transmission is sufficient for accurate control and forecasting.

In various compartmental models, the transmission rate parameter is defined as the effective contact rate, that is, the probability of infection given contact between an infectious and susceptible individual multiplied by the average rate of contacts between these groups. It is the defining rate in a disease progression and one of the two components in the basic reproductive rate, R_0, by which the continued growth or decline is decided. When $R_0 = \beta/\gamma =$ transmission rate/recovery rate < 1, an outbreak dies off; otherwise the outbreak continues to expand. The transmission rate of a disease may vary in time (take measles and influenza, for example), and models may incorporate seasonality characteristics to capture this behavior. The transmission rate may also be directly affected by social response and public health policy by which this effective contact rate is reduced to the point that the reproductive rate falls under 1, and the disease dies off. Public policies and control measures have their greatest impact on the transmission rate of a disease. Having the tools needed to recover a time-dependent transmission rate allows for the real-time analysis of the effectiveness of control measures, for the ability to determine the most powerful response and, finally, for the conceivably more accurate forecasting of the outbreak. Whereas other system parameters, i.e. incubation and recovery rates, are less dependent on intervention measures, the reproductive capacity of an outbreak and the underlying transmission rate are directly related to the efficiency of control and prevention.

There are a number of common approaches to investigating transmission rates of dis-

eases in the literature, based on system design with deterministic compartmental [56, 72–75], stochastic [76–79], and network [80–82] models being most prevalent and in many cases in combination. In these models, the common practice is to either assume a constant transmission rate [78, 82, 83], or to assume that transmission rate behaves as some pre-set periodic, exponential, or other function with a finite number of parameters [56, 77, 79, 84–86]. In recovering these parameter values, the most common methods are least squares data fitting or optimization and statistical approaches [83, 85].

In [74], Pollicott et al reconstructed a time dependent transmission rate, $\beta(t)$, by reformulating the SIR model. However their approach requires the knowledge of β_0, not easily determined, and the use of prevalence data. There are additional limitations on changes in the infected class. In [87], Hadeler modified this approach so that $S(0)$, the initial number of susceptible individuals, is assumed to be given and, though it uses incidence data, the formulation requires prevalence data at one point. Cauchemez and Ferguson [83] use a stochastic framework and MCMC to recover time-dependent transmission rate as well as other disease model parameters. The challenge here included limitations on parameter relationships and the discrete form of the recovered transmission rate.

Regardless of the type of a transmission rate, fitting model predictions for an invading pathogen to a short term incidence series results in an ill-posed problem due to instability and lack of data. For classical compartmental models, parameter identification is generally cast as an ODE constrained nonlinear optimization problem, where a numerical method has to be coupled with an appropriate regularization strategy in order to balance accuracy and stability in the reconstruction process. A reliable tool for uncertainty quantification is equally important. Even if a computational algorithm is carefully regularized, an iterative scheme for the nonlinear optimization would usually consist of solving a sequence of ill-conditioned linear equations. With noise propagation at every step, the accuracy of the recovered transmission parameters turns out to be low, especially in case of limited data for an emerging outbreak.

To partially overcome this difficulty, this chapter outlines an alternative problem-oriented approach, where the original nonlinear problem is reduced to a linear Volterra-type operator equation of the first kind. The variable transmission rate is reconstructed by fitting to both incidence and cumulative time series. Rather than pre-setting a specific shape of the unknown function by using a solution space with a small number of parameters, we discretize the time-dependent transmission rate by projecting onto a finite subspace spanned by Legendre polynomials. We further show that recovering the transmission rate as a linear combination of Legendre polynomials enables us to effectively forecast future incidence cases, the clear advantage over recovering the transmission rate at finitely many grid points within

the interval where the data is currently available.

To incorporate stability into the linear equation, we use three regularization algorithms: variational (Tikhonov's) regularization, truncated singular value decomposition (TSVD), and modified TSVD (MTSVD) [88]. The goal is to determine which stabilizing strategy is the most effective in terms of reliability of forecasting from limited data.

3.2 Problem Formulation

Consider a general SEIR transmission process [49], where the population is divided in four categories: Susceptible ($\mathcal{S}$), Exposed ($\mathcal{E}$), Symptomatic and Infectious ($\mathcal{I}$) and Removed ($\mathcal{R}$) individuals. The total population size, N, is assumed constant and initially completely susceptible to an emerging viral infection. We also assume that the population is well-mixed. That is, each individual has the same probability of having contact with any other individual in the population.

Susceptible individuals infected with a virus enter the latent period (category $\mathcal{E}$) at the rate $\beta(t)\mathcal{S}(t)\mathcal{I}(t)/N$, where $\beta(t)$ is the mean transmission rate per day (week). Latent individuals progress to the infectious class, $\mathcal{I}$, at the rate k ($1/k$ is the mean latent period). Infectious individuals recover or die at the rate γ, where the mean infectious period is $1/\gamma$. Removed, $\mathcal{R}$, are assumed to be fully protected for the duration of the outbreak. The deterministic equations of this compartmental model are given by:

$$\frac{d\mathcal{S}}{dt} = -\beta(t)\mathcal{S}(t)\frac{\mathcal{I}(t)}{N} \tag{3.1}$$

$$\frac{d\mathcal{E}}{dt} = \beta(t)\mathcal{S}(t)\frac{\mathcal{I}(t)}{N} - k\mathcal{E}(t) \tag{3.2}$$

$$\frac{d\mathcal{I}}{dt} = k\mathcal{E}(t) - \gamma\mathcal{I}(t) \tag{3.3}$$

$$\frac{d\mathcal{R}}{dt} = \gamma\mathcal{I}(t). \tag{3.4}$$

System parameters are either pre-estimated or fitted to the incidence data, $\frac{dC}{dt}$, where

$$\frac{dC}{dt} = k\mathcal{E}(t). \tag{3.5}$$

As it follows from (3.1)-(3.5),

$$\frac{d}{dt}(\mathcal{S}(t) + \mathcal{E}(t) + C(t)) = 0, \quad \mathcal{S}(t) + \mathcal{E}(t) + C(t) = \mathcal{S}(t_1) + \mathcal{E}(t_1) + C(t_1).$$

From the above, one concludes

$$\mathcal{S}(t) = -\frac{k\mathcal{E}(t)}{k} - C(t) + \mathcal{S}(t_1) + \mathcal{E}(t_1) + C(t_1) = -\frac{1}{k}\frac{dC}{dt}(t) - C(t) + \mathcal{S}(t_1) + \mathcal{E}(t_1) + C(t_1).$$

Substitute this expression into the equation

$$\frac{d\mathcal{S}}{\mathcal{S}} = -\beta(t)\frac{\mathcal{I}(t)}{N}\,dt, \quad \text{then} \quad \ln\frac{\mathcal{S}(t)}{\mathcal{S}(t_1)} = -\int_{t_1}^{t}\beta(\lambda)\frac{\mathcal{I}(\lambda)}{N}\,d\lambda. \tag{3.6}$$

To find $\mathcal{I}(t)$, use the equation

$$\frac{d\mathcal{I}}{dt} + \gamma\mathcal{I}(t) = \frac{dC}{dt},$$

which gives

$$\mathcal{I}(t) = \mathcal{I}(t_1)\exp(-\gamma(t-t_1)) + \int_{t_1}^{t}\exp(-\gamma(t-\lambda))\frac{dC}{d\lambda}(\lambda)\,d\lambda. \tag{3.7}$$

Combining (3.6) and (3.7), one derives

$$-N\ln\left[\frac{-\frac{1}{k}\frac{dC}{dt}(t) - C(t) + \mathcal{E}(t_1) + C(t_1)}{\mathcal{S}(t_1)} + 1\right]$$

$$= \int_{t_1}^{t}\beta(\tau)\left\{\mathcal{I}(t_1)\exp(-\gamma(\tau-t_1)) + \int_{t_1}^{\tau}\exp(-\gamma(\tau-\lambda))\frac{dC}{d\lambda}(\lambda)d\lambda\right\}d\tau, \tag{3.8}$$

which is a linear Volterra-type integral equation of the first kind with the unknown transmission rate, $\beta(t)$, to be recovered from limited cumulative and incidence time series, $C(t)$ and $\frac{dC}{dt}$, respectively.

3.3 Regularization Strategies and Discrete Approximation

As it has been established in the previous section, the reconstruction of $\beta(t)$ can be reformulated as a linear equation of the first kind with noise added to the response vector and to the operator itself

$$\mathcal{A}\beta = g, \quad \mathcal{X} \to \mathbb{R}^m, \tag{3.9}$$

where $\mathcal{A}$ is given by its h-approximation, $||\mathcal{A}-\mathcal{A}_h|| \leq h$, and g is given by its δ-approximation, $||g - g_\delta|| \leq \delta$. Noise enters the operator through incidence data under the kernel:

$$\mathcal{A}_h\beta(t) := \int_{t_1}^{t} \beta(\tau) \left\{ \mathcal{I}(t_1) \exp(-\gamma(\tau - t_1)) + \int_{t_1}^{\tau} \exp(-\gamma(\tau - \lambda)) \frac{dC}{d\lambda}(\lambda) d\lambda \right\} d\tau,$$

and it enters the right-hand side through its dependence on both incidence and cumulative data as shown in (3.8):

$$g_\delta := -N \ln \left[\frac{-\frac{1}{k}\frac{dC}{dt}(t) - C(t) + \mathcal{E}(t_1) + C(t_1)}{\mathcal{S}(t_1)} + 1 \right].$$

The true solution, $\beta(t)$, in (3.9) lies in a Hilbert space $\mathcal{X}$, the noise-contaminated operator, $\mathcal{A}_h$, maps $\mathcal{X}$ into $\mathbb{R}^m$, and g_δ is a vector in the finite-dimensional data space, $\mathbb{R}^m$. Due to the nature of our application, $\mathcal{X}$ is infinite dimensional and, upon discretization, its dimensionality is much larger than m. This results in an ill-posed problem that needs to be regularized prior to its inversion.

To introduce the proposed regularization strategies, we consider a singular system of the operator $\mathcal{A}_h$, $\{u_i, \sigma_i, v_i\}_{i=1}^m$, with singular values

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m > 0.$$

Here $v_i \in \mathcal{X}$ and $u_i \in \mathbb{R}^m$ are such that $(v_i, v_j)_{\mathcal{X}} = \delta_{ij}$ and $(u_i, u_j)_{\mathbb{R}^m} = u_i^T u_j = \delta_{ij}$, and δ_{ij} denotes the Kronecker delta, equal to 1 if $i = j$ and to 0 otherwise. If a regularized solution, β_α, is obtained by filtering the noisy data

$$\beta_\alpha := \sum_{i=1}^{m} \omega_\alpha(\sigma_i) \frac{(u_i, g_\delta)}{\sigma_i} v_i := R_{\alpha,h} g_\delta, \tag{3.10}$$

then the choice of ω_α defines a particular type of a regularization strategy $R_\alpha : \mathbb{R}^m \to \mathcal{X}$, and α is a stabilizing parameter, which regulates the extent of filtering and depends on the level of noise in $\mathcal{A}_h$ and g_δ. To reconstruct time-dependent transmission rate, $\beta(t)$, we use three admissible filters, which ensure convergence of the regularized solution as the noise level tends to zero, [6]:

1) $\omega_\alpha(\sigma) = \frac{\sigma^2}{\sigma^2+\alpha}$ (Tikhonov's regularization),

$$2)\ \omega_\alpha(\sigma) = \begin{cases} 1, & \sigma \geq \sqrt{\alpha} \\ 0, & \sigma < \sqrt{\alpha} \end{cases} \quad \text{(Truncated SVD)},$$

$$3)\ \omega_\alpha(\sigma) = \begin{cases} 1, & \sigma \geq \sqrt{\alpha} \\ \frac{\sigma}{\sqrt{\alpha}}, & \sigma < \sqrt{\alpha} \end{cases} \quad \text{(Modified Truncated SVD)}.$$

The first two filters are probably the most known and the most used. The third filter (MTSVD) was recently studied in [88] for the case of a noise-free operator. It has a remarkable optimal property: *among all filters with the same level of stability, it provides the highest accuracy of the algorithm.* In a subsequent section, we will verify this property for the case of noise present both in the operator and in the right-hand side. In our numerical experiments, discussed in the next sections, all three filters give consistent results. However, MTSVD tends to do slightly better in terms of forecasting from limited data.

As we discretize the unknown transmission rate, $\beta(t)$, our goal is not to incorporate any specific behavior of this function in equation (3.9). Instead, we attempt to *recover* that behavior in addition to recovering numerical values for all entries of the solution vector. To that end, we project $\beta(t)$ onto a finite subset spanned by the shifted Legendre polynomials of degree $0, 1, ..., n$ defined recursively as follows

$$x = \frac{2t - a - b}{b - a}, \qquad P_0(x) = 1, \qquad P_1(x) = x, \qquad t \in [a, b],$$

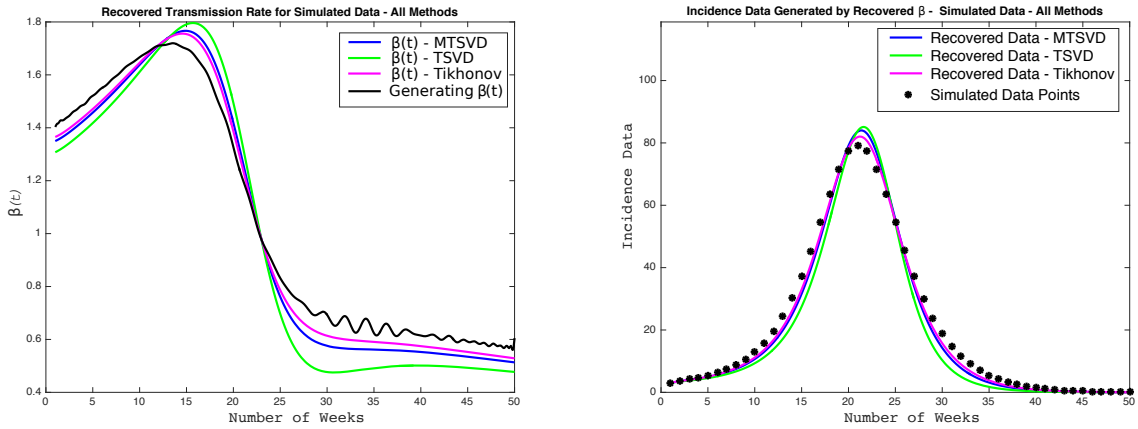
$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x).$$

These functions are orthogonal on the interval $[a, b]$, the duration of the outbreak, with respect to the $\mathcal{L}_2$ inner product

$$(P_n, P_k) \ = \ \frac{b-a}{2n+1}\,\delta_{nk}.$$

The discretization of the original operator equation by projection onto a finite subspace spanned by Legendre polynomials results in solving (in the sense of least squares) a linear system of m equations with $n+1$ unknowns for the coefficients $\mathcal{C}_i$, $i = 0, 1, ..., n$, in the Legendre polynomial expansion and then computing $\beta_\alpha(t)$ as

$$\beta_\alpha(t) = \sum_{i=0}^{n} \mathcal{C}_i P_i(t), \quad t \in [a, b].$$



(a) $\beta(t)$ (b) *Incidence Case Curve*

Figure 3.1. *Recovery of* $\beta(t)$ *and Incidence Data for Simulated Data - Full Data Set*

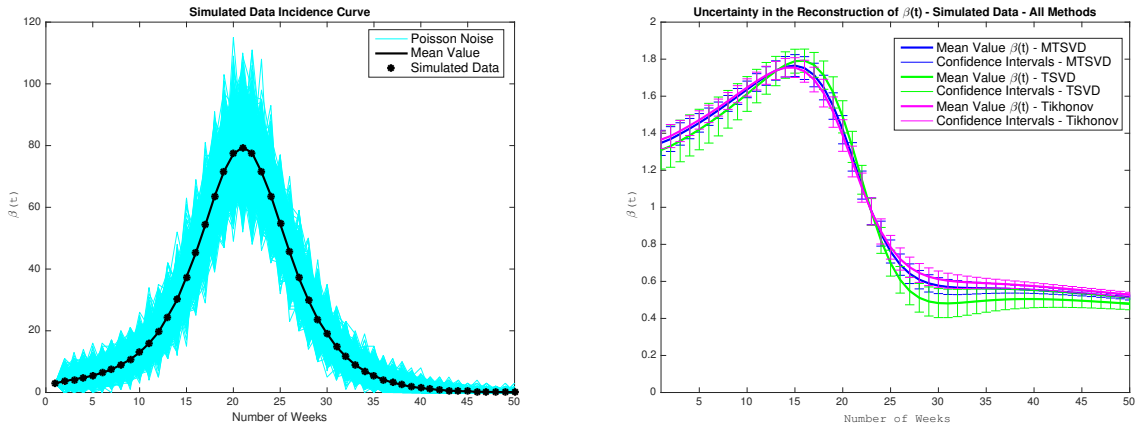
For all three regularization algorithms, the value of α is chosen from the goodness of fit of the incidence data, generated by β_α, to the real data used for the inversion (discrepancy principle).

3.4 Numerical Experiments with Simulated Data

First, we test the above regularization methods using a simulated set of data. The experiment is conducted as follows. We discretize the infinite dimensional Hilbert space, $\mathcal{X} = \mathcal{L}_2[a, b]$, by projecting onto a subspace spanned by a large number of Legendre polynomials (250) to get an accurate approximation of the original $\beta(t)$. Given this $\beta(t)$, we generate incidence data by solving the forward problem, Figure 3.1. Once the incidence data, $\frac{dC}{dt}$, have been computed, we solve the inverse problem by TSVD, MTSVD, and Tikhonov's regularization algorithms. To examine both regularization and discretization errors, while solving the inverse problem we discretize $\mathcal{X}$ with a smaller number of Legendre polynomials (100). Figure 3.1 illustrates how the original $\beta(t)$ compares to $\beta_\alpha(t)$ recovered by each regularization scheme.

In (3.8), we choose the total population size, N, and the initial number of cumulative cases, $C(0)$, to be 1.5×10^6 and 3, respectively. Mimicking an 8 day latent period and a 6 day infectious period, we take $\kappa = 7/8$ and $\gamma = 7/6$, and assume that $[a, b] = [1, 50]$, i.e., the outbreak is over in 50 weeks. Due to ill-posedness of the inverse problem, all three regularization methods are semi-convergent in a sense that the discrepancy initially goes down as α decreases, but then it begins to grow after α reaches a certain admissible level. We choose α right before it happens. For that α, the discrepancy is about the same as the

amount of noise in the incidence data. The values of the regularization parameter, α, as selected by the discrepancy principle [89], are 1.26×10^{-7}, 3.00×10^{-8}, and 1.50×10^{-8} for MTSVD, TSVD and Tikhonov regularization methods.



(a) *Poisson Curves* (b) *Uncertainty in $\beta_\alpha(t)$ - All Methods*

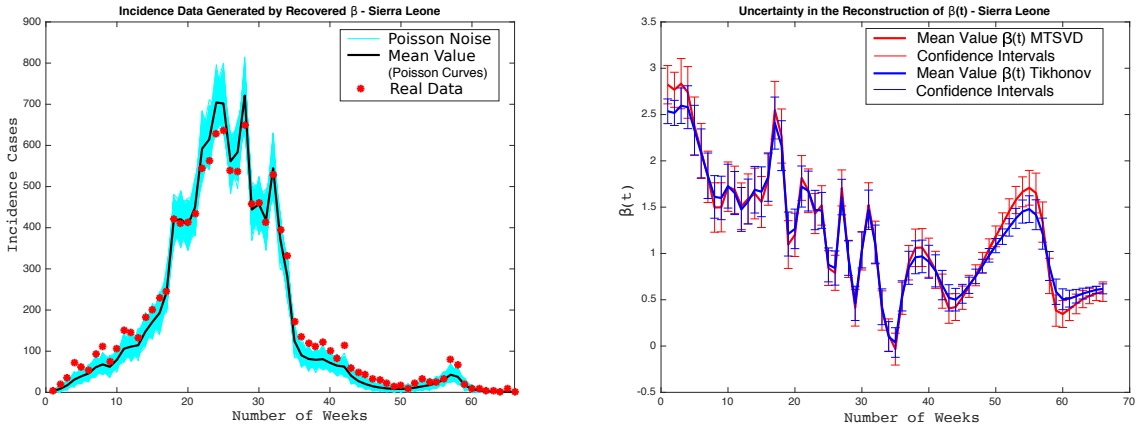
Figure 3.2. *Noisy Data Used to Quantify Uncertainty in $\beta_\alpha(t)$ - Simulated Data*

To quantify uncertainty in $\beta_\alpha(t)$ for the three methods, we take the simulated incidence curve (shown in black in Figure 3.1), add Poisson noise to this curve 2000 times, and reconstruct the mean value of $\beta_\alpha(t)$ and the associated 95% confidence intervals. Figure 3.2 gives the Poisson curves and the results for all methods on a single plot. We note that all three methods produce similar reconstructions and that of the methods, TSVD results in the most variation from the generating transmission rate.

3.5 Approximation of Time-Dependent Transmission Rate and Quantification of Uncertainty for Real Data

In this section, we use real data for the most recent outbreak of Ebola Virus Disease (EVD) in West Africa, predominately affecting Guinea, Liberia, and Sierra Leone [1], in order to examine regularizing properties of the proposed algorithms.

In the beginning of our numerical analysis, we take full data set for the 2014 EVD outbreak in Sierra Leone and apply Tikhonov's, TSVD, and MTSVD regularization schemes to pre-estimate $\beta_\alpha(t)$ in each case. Using this initially recovered $\beta_\alpha(t)$, we generate the incidence curve, add Poisson noise, 2000 iterations for the results given, and reconstruct the corresponding $\beta_\alpha(t)$ via the respective methods. This yields 95% confidence intervals for the

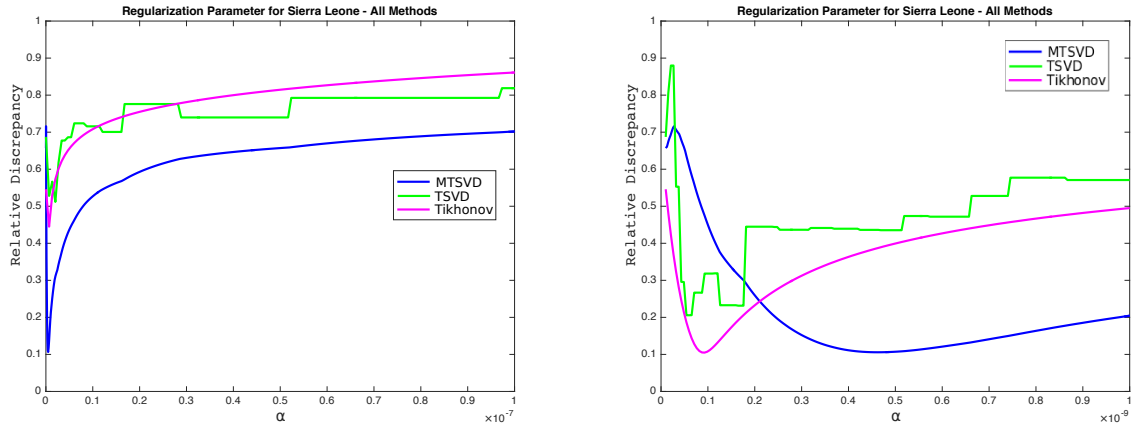


(a) *MTSVD Data* (b) *Recovered* $\beta_\alpha(t)$

Figure 3.3. *Noisy Data Used to Quantify Uncertainty in* $\beta_\alpha(t)$ *- Sierra Leone - Full Data Set*

approximate $\beta_\alpha(t)$ as well as the mean values of the recovered function. The reconstructed values of $\beta_\alpha(t)$ and the corresponding forecasting curves for TSVD and Tikhonov's regularization schemes are very difficult to tell apart. Therefore TSVD results are not included in Figure 3.3. Tikhonov's and MTSVD approximations of $\beta_\alpha(t)$ *are* slightly different, Figure 3.3.

The forecasting curves for partial data sets obtained with MTSVD $\beta_\alpha(t)$ are the most accurate and the least uncertain as illustrated in Figures 3.5 and 3.7 below.



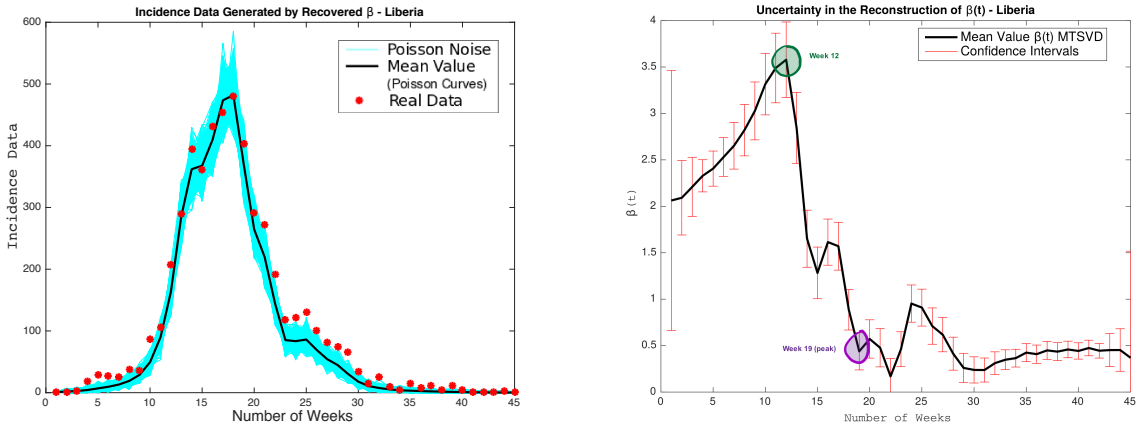
(a) *Global Picture* (b) *Local Behavior*

Figure 3.4. *Selection of Regularization Parameter - Sierra Leone - Full Data Set*

Figure 3.4 demonstrates the parameter selection process for this experiment. The first

plot in Figure 3.4 shows the dependence of relative discrepancy on α in the interval $[0, 10^{-7}]$. For each method, the corresponding graph gives the lower bound of α that cannot be crossed. If α moves below this value, the discrepancy goes up almost vertically, and the relative error on the generated data quickly reaches 100%. The TSVD curve illustrates the discrete nature of TSVD regularization: for all values of α between two consecutive singular values, σ_k and σ_{k+1}, the filtering function, ω_α, remains the same, and therefore the regularized solution, $\beta_\alpha(t)$, and the resulting discrepancy do not change either. After the initial preview of a big picture, we magnify the area where the discrepancy reaches its minimum, $[0, 10^{-9}]$, and for each method we select the smallest value of α where this minimum is attained.

The reproduction number, R_0, of an outbreak gives the number of cases one case generates on average over the course of its infectious period. When $R_0 < 1$, we expect the outbreak to die out; with $R_0 > 1$, the infection can spread and with higher values of R_0, it can become harder to control the outbreak. In the model given, $R_0(t) = \beta(t)/\gamma$ and therefore reconstruction of time-dependent $\beta(t)$ has direct ties to $R_0(t)$. Since $R_0 \propto \beta$, qualitatively the behaviors are the same. The transmission rate and the corresponding $R_0(t)$ curve, recovered from Sierra Leone data, evidence sporadic decline (Figure 3.3). Some of this behavior may be attributed to noise in the data, but for the most part we see it as the result of less than effective implementation of control measures.



(a) *Poisson Incidence Curves* (b) $\beta(t)$

Figure 3.5. *Uncertainty Quantification for the Recovered $\beta_\alpha(t)$ - Liberia - Full Data Set*

When we apply the MTSVD regularization method to Liberia's data from the 2014 EVD outbreak using the full data set, we see a marked drop in the transmission rate, Figure 3.5, and a more smooth transition to an outbreak die off level. The application of MTSVD enables us to capture differing behaviors of the transmission rate that may be correlated

to the efficiency of control measures or other intervention tools impacting the transmission rate.

3.6 Forecasting from Limited Data for Emerging Outbreaks

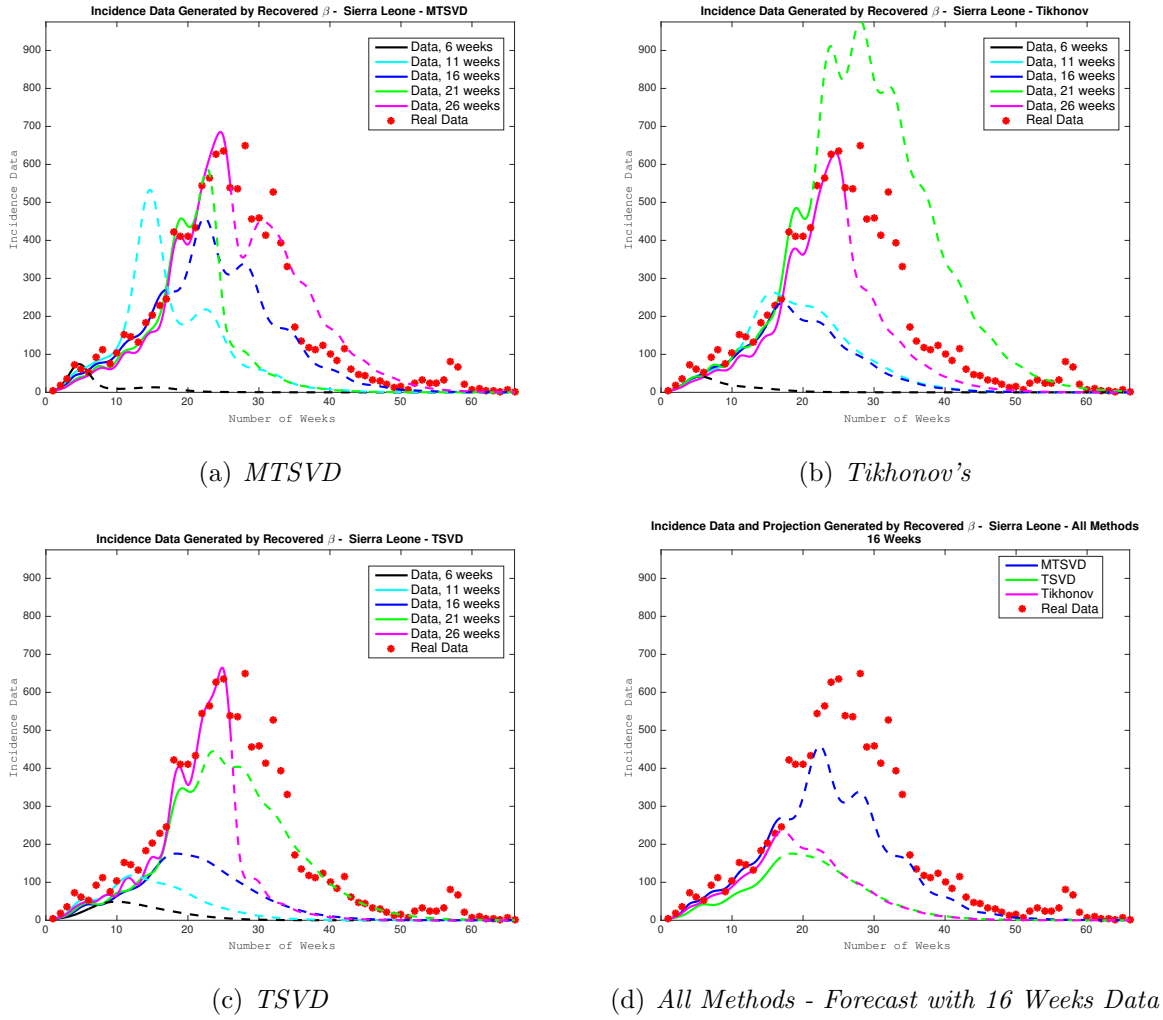
Since our algorithm produces coefficients in the Legendre polynomial expansion, we can use $\beta_\alpha(t)$ recovered from early data to forecast the remaining part of the outbreak. In order to determine the forecasting curves, data is taken for the first m weeks and the regularization parameter, α_m, is estimated by the discrepancy principle [89] as in the previous sections. At the next step, $\beta_\alpha(t)$ is recovered based on m weeks of data. It is then used to generate an incidence curve for the entire duration of the outbreak.

Table (3.1) *Regularization Parameters Chosen By Discrepancy - Sierra Leone*

	MTSVD		TSVD		Tikhonov	
Week	α	RD	α	RD	α	RD
6	6.31e-11	0.123	6.12e-11	0.692	6.12e-11	0.440
11	1.04e-10	0.174	6.12e-11	0.316	6.12e-11	0.322
16	1.83e-10	0.139	6.12e-11	0.427	6.12e-11	0.198
21	3.67e-10	0.130	1.22e-10	0.263	6.12e-11	0.146
26	5.07e-10	0.101	6.12e-11	0.107	1.22e-10	0.122

In the first forecasting experiment, we employ all three, TSVD, Tikhonov's, and MTSVD regularization methods on limited data sets for 2014 EVD outbreak in Sierra Leone [1]. Table 3.1 gives the respective chosen regularization parameters for each method and the associated relative discrepancy (RD). For Sierra Leone, MTSVD does a better job forecasting from 6 and 26 weeks of incidence data, Figure 3.6 (a). The results from using Tikhonov's method tend to either significantly understate or overstate the forecasted incidence until after the outbreak's peak, Figure 3.6 (b). The results obtained by TSVD algorithm tend to consistently underestimate future incidence cases. In Figure 3.6 (d) we show the forecasting results for all three methods using 16 weeks of data; MTSVD clearly outperforms.

While MTSVD does a better job at forecasting with time-dependent $\beta(t)$, all three methods are a vast improvement over forecasting that results from the use of constant β. We compare them in Figure 3.7. The forecasting curves for Liberia (with a time-dependent $\beta(t)$) at week 13 indicate a potentially much larger outbreak; the largest recovered reproductive number was observed at week 12. This is not surprising if one takes into consideration that $\beta(t)$ is growing for the first 12 weeks, when the outbreak is on its rise. However, between



(a) *MTSVD* (b) *Tikhonov's*

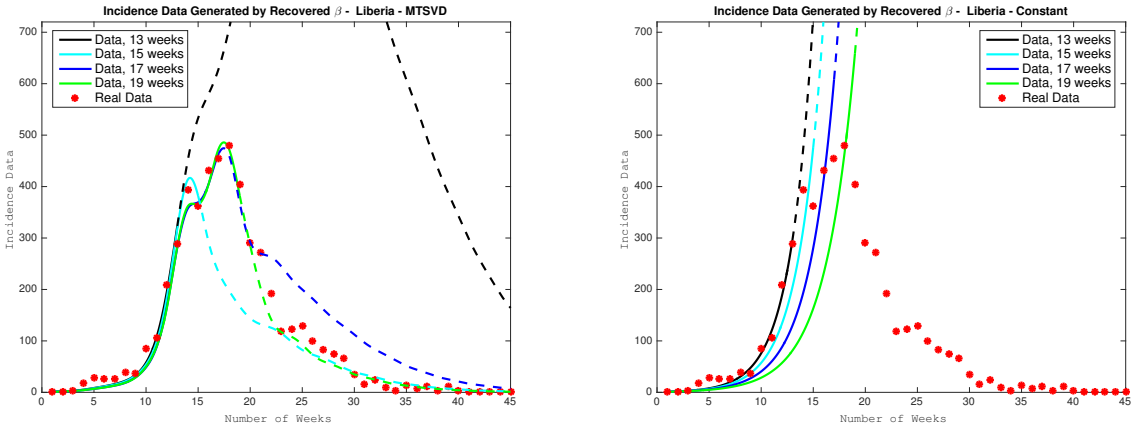
(c) *TSVD* (d) *All Methods - Forecast with 16 Weeks Data*

Figure 3.6. *Comparison of Forecasting Curves Using Partial Data - Sierra Leone*

weeks 12 and 13, $\beta(t)$ declines very quickly. The forecasting curve captures that decline and, despite of overestimating future cases, it shows a clear turning point (that is not far from the actual turning point), and a rapid decrease afterwards. Table 3.2 gives the comparisons of projected incidence cases using MTSVD and constant β showing case counts projected (actual cases are in parentheses) for each method.

The subsequent forecasting curves with a time-dependent $\beta(t)$ do an excellent job in approximating future incidence levels. The forecasting curves with a constant β show a growing number of incidence cases suggesting the growth will continue until the population runs out of susceptible individuals. Additional forecasting curves for various districts for the EVD outbreak are given at the end of this chapter.

The next experiment shows that one can use early data, before incidence peak, to forecast in short forward time the projected incidence cases with confidence intervals. In



(a) *Variable* $\beta(t)$ (b) *Constant* $\beta(t)$

Figure 3.7. *Forecasting Curves Using Partial Data - Liberia - MTSVD*

Table (3.2) *Comparison of Forecasting - MTSVD and Constant* β

# of Partial Weeks Data	Projected (Actual) Incidence					
	MTSVD			Constant β		
Week	# Weeks Forward			# Weeks Forward		
(Actual Incidence)	4	5	6	4	5	6
13 (289)	632 (454)	715 (480)	816 (404)	1814 (454)	2852 (480)	4478 (404)
15 (362)	169 (404)	147 (291)	136 (272)	2588 (404)	3944 (291)	6005 (272)
17 (454)	259 (272)	251 (192)	232 (118)	2849 (272)	4184 (192)	6138 (118)
19 (404)	114 (118)	105 (122)	90 (130)	2648 (118)	3746 (122)	5293 (130)

this application, we utilize m weeks of data and recover $\beta_\alpha(t)$. Given this $\beta_\alpha(t)$, we generate the initial m week incidence data curve and add Poisson noise to this curve (2000 iterations for the results given). For each noisy curve, $\beta_\alpha(t)$ is recovered employing a data-specific regularization parameter, α. Each recovered $\beta_\alpha(t)$ is then used to project forward $m+5$ weeks for Sierra Leone and $m+2$ weeks for Liberia, and confidence intervals are determined from the forecasts at each week. We repeat this process every 5 and 2 weeks, respectively, until incidence peak is reached, and present the results for Sierra Leone and Liberia in Figure 3.8 (a) and (b).

Forecasting for Sierra Leone begins at week 11 and does an excellent job capturing future epidemic behavior. For Liberia, where we begin forecasting at week 12, there tends to be an overestimate of incidence cases. This can be explained when we consider the behavior of $\beta_\alpha(t)$ in Figure 3.5. We note that the peak of $R_0(t)$ occurs at week 12, and the reproduction rate makes a sharp drop continuing to epidemic peak at week 19, and this is the period of forecasting. The overestimate in this method is considerably less significant when compared

to either a constant β forecast or to forecasting with Tikhonov's and TSVD regularization methods.

The impact of intervention and control on a disease transmission rate can also be seen when the algorithm is applied to outbreaks other than EVD. The recovered transmission rate may then be used to forecast future incidence cases. Prior to the implementation of vaccination for measles (1948-1964), outbreaks of the disease were common. The 1948 outbreak in London produced $28,000+$ cases in 40 weeks [1]. The model parameters are $\kappa = 7/8$ and $\gamma = 7/6$ indicating an 8 day latent period and a 6 day infectious period. London's population in 1948 was 8,200,000.

Another example is the pandemic influenza outbreak in 1918, which affected many cities. San Francisco experienced 28,310 cases in 63 days [1]. For this disease the latent and infectious periods are given as 2 and 4 days respectively; the population of San Francisco at that time was 550,000. Figure 3.8 (c) and (d) demonstrate forecasting results for the 1948 measles outbreak in London and for the 1918 pandemic influenza outbreak in San Francisco obtained with MTSVD regularization method.

3.7 Theoretical Analysis

In what follows we show that gentle regularization

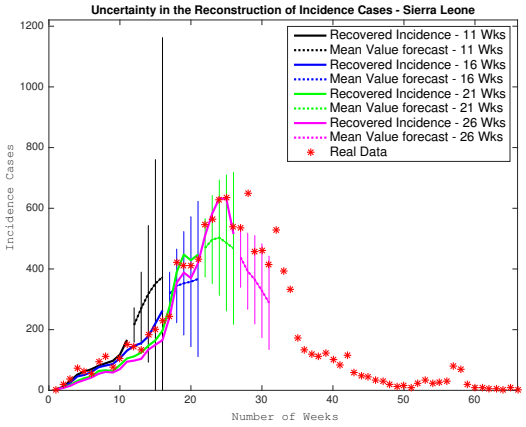
$$\beta_\alpha := \sum_{i=1}^{n} \omega_\alpha(\sigma_i) \frac{(u_i, g_\delta)}{\sigma_i} v_i := R_{\alpha,h} g_\delta \quad \text{with} \quad \omega_\alpha(\sigma) = \begin{cases} 1, & \sigma \geq \sqrt{\alpha} \\ \frac{\sigma}{\sqrt{\alpha}}, & \sigma < \sqrt{\alpha} \end{cases} \tag{3.11}$$

has a certain optimal property, which may be the reason for some computational advantage it has over other regularization algorithms. Let $\hat{\beta}$ be the exact solution to $\mathcal{A}\beta = g$. From (3.11), one concludes

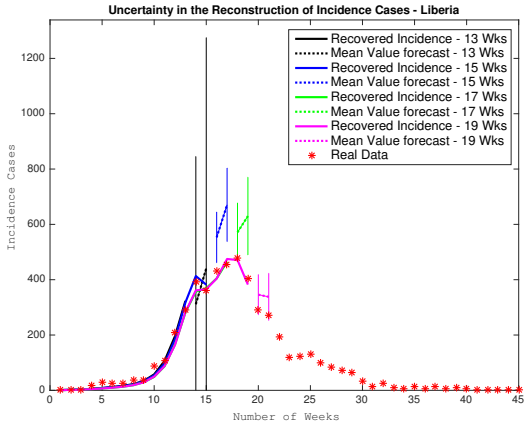
$$\hat{\beta} - \beta_\alpha = \hat{\beta} - R_{\alpha,h}\mathcal{A}_h\hat{\beta} + R_{\alpha,h}[(\mathcal{A}_h - \mathcal{A})\hat{\beta} + g - g_\delta],$$

and therefore

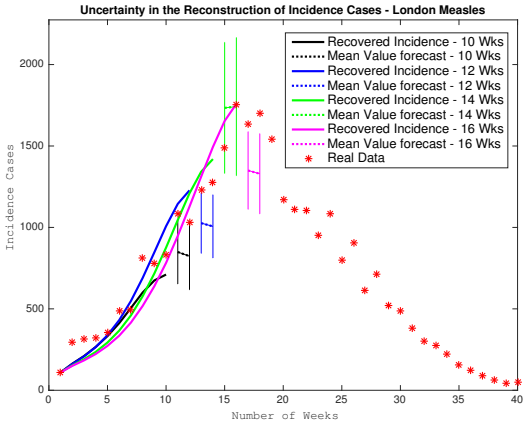
$$||\hat{\beta} - \beta_\alpha|| \leq ||\hat{\beta} - R_{\alpha,h}\mathcal{A}_h\hat{\beta}|| + ||R_{\alpha,h}||\,||\mathcal{A}|| \left[\frac{||\mathcal{A}_h - \mathcal{A}||\,||\hat{\beta}||}{||\mathcal{A}||} + \frac{||g - g_\delta||}{||\mathcal{A}||} \right].$$



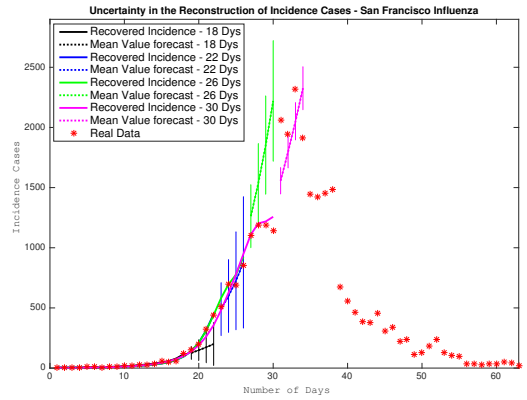
(a) *Sierra Leone - EVD*



(b) *Liberia - EVD*



(c) *London Measles Outbreak*



(d) *San Francisco Influenza*

Figure 3.8. *Short Term Forecasting Curves - With Confidence Intervals - MTSVD*

Taking into consideration the obvious estimate $||g|| \leq ||\mathcal{A}||\,||\hat{\beta}||$, i.e., $\frac{1}{||\mathcal{A}||} \leq \frac{||\hat{\beta}||}{||g||}$, one obtains

$$\frac{||\hat{\beta} - \beta_\alpha||}{||\hat{\beta}||} \leq \frac{||\hat{\beta} - R_{\alpha,h}\mathcal{A}_h\hat{\beta}||}{||\hat{\beta}||} + \underbrace{||R_{\alpha,h}||\,||\mathcal{A}||}_{\text{cond}_{R_{\alpha,h}}(\mathcal{A})}\left[\frac{||\mathcal{A}_h - \mathcal{A}||}{||\mathcal{A}||} + \frac{||g - g_\delta||}{||g||}\right]. \tag{3.12}$$

In case of a noise-free operator, the first term in (3.12) measures the loss of accuracy due to a numerical algorithm, and if one passes to the limit in (3.12) as $\alpha \to 0^+$, one gets the classical estimate

$$\frac{||\hat{\beta} - \beta_\delta||}{||\hat{\beta}||} \leq \underbrace{||\mathcal{A}^{-1}||\,||\mathcal{A}||}_{\text{cond}(\mathcal{A})}\frac{||g - g_\delta||}{||g||} \tag{3.13}$$

known for un-regularized problems. Hence the product $||R_{\alpha,h}||\,||\mathcal{A}||$ may be understood as the generalized condition number.

Now assuming that both the operator, $\mathcal{A}$, and the right-hand side, g, are noise contaminated, i.e., $||\mathcal{A}_h - \mathcal{A}|| \le h$ and $||g_\delta - g|| \le \delta$, we formulate the problem: *among all regularizing strategies with the same* $cond_{R_{\alpha,h}}(\mathcal{A}) = ||R_{\alpha,h}||\,||\mathcal{A}||$, *find a strategy that minimizes the error of the computational algorithm,* $||\hat{\beta} - R_{\alpha,h}\mathcal{A}_h\hat{\beta}||/||\hat{\beta}||$.

Let $\mathcal{N}$ be the fixed value of the generalized condition number $cond_{R_{\alpha,h}}(\mathcal{A})$. It appears that gentle truncation (3.11) solves the above problem provided that the regularization parameter, $\hat{\alpha}$, is selected as $\hat{\alpha} = \frac{||\mathcal{A}||^2}{\mathcal{N}^2}$. In other words,

$$\beta_{opt} := \sum_{i=1}^{n} \hat{\omega}_{\hat{\alpha}}(\sigma_i)\frac{(u_i, g_\delta)}{\sigma_i}v_i := \hat{R}_{\hat{\alpha},h}g_\delta, \tag{3.14}$$

where

$$\hat{\omega}_{\hat{\alpha}}(\sigma) = \begin{cases} 1, & \sigma \ge ||\mathcal{A}||/\mathcal{N} \\ \mathcal{N}\sigma/||\mathcal{A}||, & \sigma < ||\mathcal{A}||/\mathcal{N}. \end{cases} \tag{3.15}$$

Indeed, assume the converse. From (3.11) and (3.12), one has

$$||\hat{\beta} - R_{\alpha,h}\mathcal{A}_h\hat{\beta}|| = \left\|\hat{\beta} - \sum_{i=1}^{n}\omega_\alpha(\sigma_i)\frac{(u_i, \mathcal{A}_h\hat{\beta})}{\sigma_i}v_i\right\|. \tag{3.16}$$

Notice that

$$(u_i, \mathcal{A}_h\hat{\beta}) = (\mathcal{A}_h^* u_i, \hat{\beta}) = \sigma_i(v_i, \hat{\beta}). \tag{3.17}$$

According to (3.16) and (3.17), one concludes

$$||\hat{\beta} - R_{\alpha,h}\mathcal{A}_h\hat{\beta}||^2 = \left\|\hat{\beta} - \sum_{i=1}^{n}\omega_\alpha(\sigma_i)(v_i, \hat{\beta})v_i\right\|^2 = \sum_{i=1}^{n}(1 - \omega_\alpha(\sigma_i))^2|(v_i, \hat{\beta})|^2.$$

Let $\bar{R}_{\bar{\alpha},h} := \sum_{i=1}^{n}\bar{\omega}_{\bar{\alpha}}(\sigma_i)\frac{(u_i,\cdot)}{\sigma_i}v_i$ be some other strategy with $cond_{R_{\alpha,h}}(\mathcal{A}) = \mathcal{N}$ that results in a higher accuracy of the algorithm as compared to $\hat{R}_{\hat{\alpha},h}$. Then there exists $\sigma_j,\ 0 < \sigma_j \le ||\mathcal{A}_h||$, such that

$$(1 - \bar{\omega}_{\bar{\alpha}}(\sigma_j))^2 < (1 - \hat{\omega}_{\hat{\alpha}}(\sigma_j))^2. \tag{3.18}$$

Since $0 \leq \omega_\alpha(\sigma) \leq 1$ for all $\alpha > 0$ and $0 < \sigma \leq ||\mathcal{A}_h||$, (3.15) and (3.18) imply

$$1 - \bar{\omega}_{\bar{\alpha}}(\sigma_j) < \begin{cases} 0, & \sigma_j \geq ||\mathcal{A}||/\mathcal{N} \\ 1 - \mathcal{N}\sigma_j/||\mathcal{A}||, & \sigma_j < ||\mathcal{A}||/\mathcal{N}. \end{cases}$$

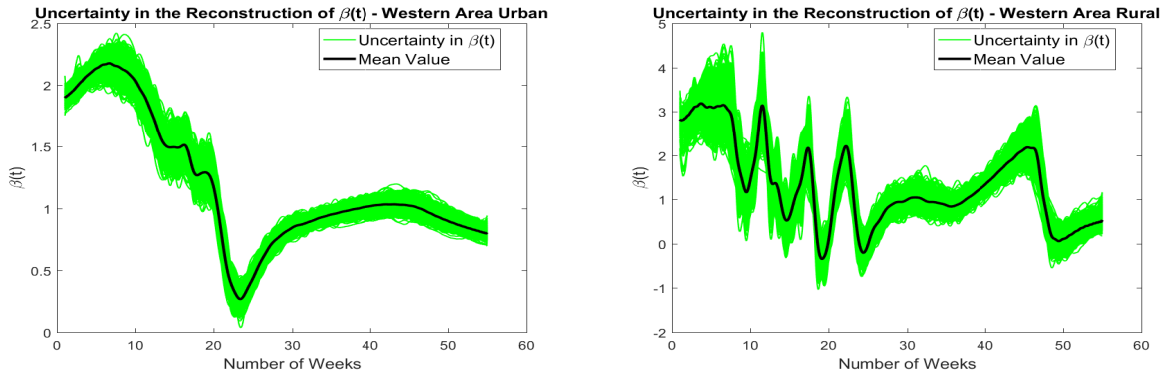
The first case is not possible. The second case yields $\bar{\omega}_{\bar{\alpha}}(\sigma_j) > \mathcal{N}\sigma_j/||\mathcal{A}||$. Hence

$$||\bar{R}_{\bar{\alpha},h}u_j||^2 = \left\| \sum_{i=1}^{n} \bar{\omega}_{\bar{\alpha}}(\sigma_i) \frac{(u_i, u_j)}{\sigma_i} v_i \right\|^2 = \left[\frac{\bar{\omega}_{\bar{\alpha}}(\sigma_j)}{\sigma_j} \right]^2 |(u_j, u_j)|^2 > \frac{\mathcal{N}^2}{||\mathcal{A}||^2} ||u_j||^2 = \frac{\mathcal{N}^2}{||\mathcal{A}||^2},$$

which proves that $cond_{\bar{R}_{\bar{\alpha},h}}(\mathcal{A}) = ||\bar{R}_{\bar{\alpha},h}||\,||\mathcal{A}|| > \mathcal{N}$. Thus we arrive at a contradiction, and the choice of $\hat{R}_{\hat{\alpha},h}$ by (3.14)-(3.15) is, in fact, optimal.

3.8 Numerical Results for Additional Data Sets

In this section we employ data subsets to illustrate both stable parameter estimation and future projections that can be obtained utilizing our methods. The use of national data sets and the algorithm presented in this chapter have been proven to generate reliable short term forecasts using limited (early) data for an ongoing outbreak. For many diseases a number of different partitions of the data can be made by age, sex, income level or geography. In the case of the 2014 EVD outbreak, given data for various geographic districts provided by Dr. G. Chowell, one can apply the proposed method to a specific subset

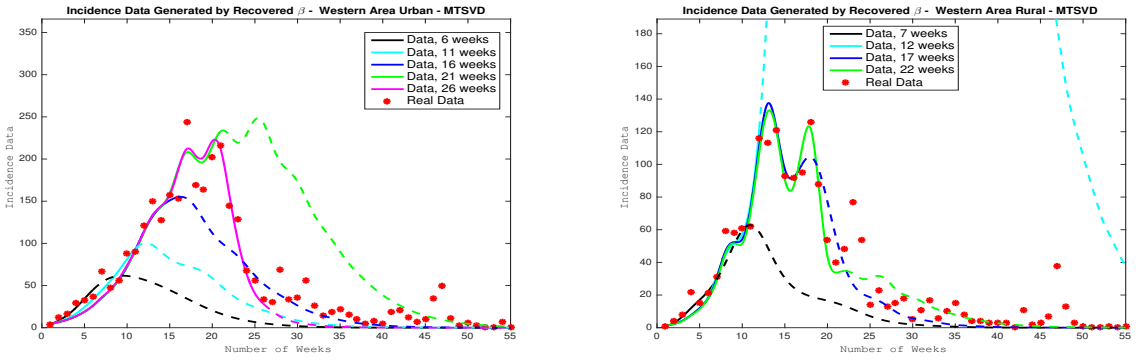


(a) *Western Area Urban, Sierra Leone,* $\beta(t)$ (b) *Western Area Rural, Sierra Leone,* $\beta(t)$

Figure 3.9. *Uncertainty Quantification for the Transmission Rate Recovered from Full Data*

of a more general outbreak. In what follows we show the efficiency of our algorithm for pairs of districts from Sierra Leone and Liberia such that there is sharp differences in population densities and access to medical care. These additional numerical studies further establish the practicality of the approach within strongly differing underlying demographics.

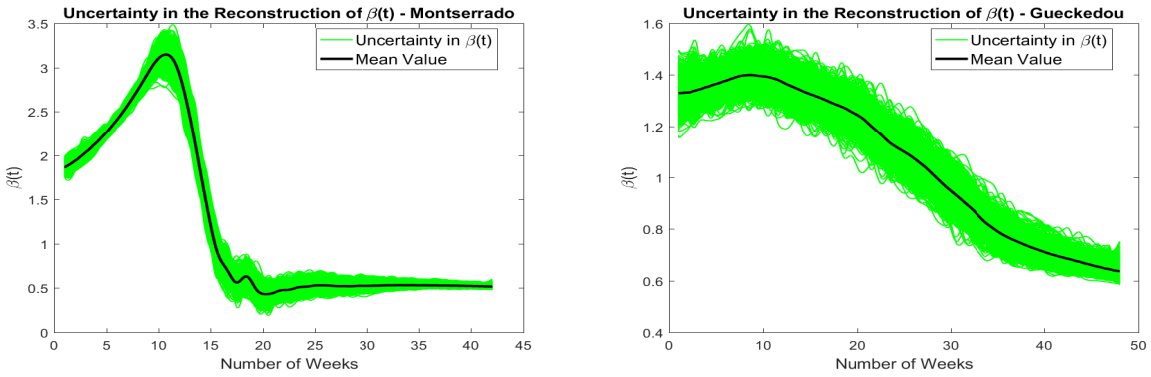
Western Area Urban and Western Area Rural are 2 of 14 districts in Sierra Leone. Freetown, the capital and the largest city in Sierra Leone, is located in Western Area Urban. Population for the two districts are 1.1 million and 500 thousand, respectively, and these two districts were primary hotspots of the 2014 EVD outbreak. Utilizing weekly data sets from these two regions [1], we approximate $\beta_\alpha(t)$ by MTSVD regularization method. Figure 3.9 gives these results. We use 2000 noisy data sets to quantify the uncertainty. It appears that the transmission rate was effectively reduced in the urban district, but that this reduction was more erratic in the rural area. The plots of both transmission rates and forecasted incidence suggest a less than effective mitigation of the transmission rate, especially in rural areas. In the rural area, the decline in speed and behavior more closely matches the country-wide result. The forecasting curves for both districts are given in Figure 3.10.



(a) *Western Area Urban - Sierra Leone* (b) *Western Area Rural - Sierra Leone*

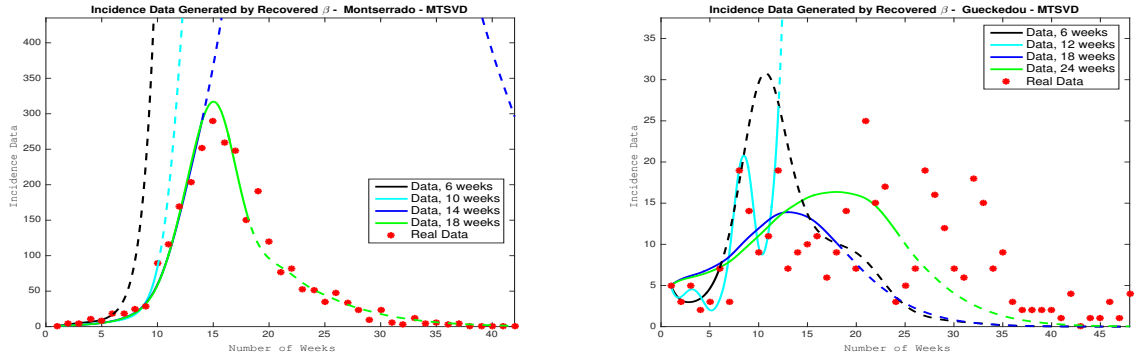
Figure 3.10. *Comparison of Forecasting Results for the Recovered* $\beta_\alpha(t)$

The 2014 outbreak in Liberia consisted of 10,678 cases and 4810 deaths. The numerical experiments are conducted with the country wide data for the outbreak as well as for two of the country's districts: Montserrado and Gueckedou [1]. Montserrado district is home to Monrovia, the capital of Liberia, and two of its infected individuals were responsible for the outbreak in Nigeria and for the cases in the United States. Gueckedou is the site of the index case for the 2014 outbreak and is located in the vicinity of the conflux of borders between Liberia, Sierra Leone and Guinea. Figure 3.11 give the uncertainty quantification for 2000



(a) *Montserrado District, Liberia,* $\beta(t)$ (b) *Gueckedou District, Liberia,* $\beta(t)$

Figure 3.11. *Uncertainty Quantification for the Transmission Rate Recovered from Full Data*



(a) *Montserrado District* (b) *Gueckedou District*

Figure 3.12. *Comparison of Forecasting Results for the Recovered* $\beta_\alpha(t)$

noisy curves. Figure 3.12 illustrates the forecasting results. For these Liberia districts, we see similarity in the behavior of $\beta_\alpha(t)$ for the urban district of Montserrado as compared to the country as a whole. The rural district exhibits a slower decline in the transmission rate, though from a lower level. The sharp decline in transmission rate and the forecasted curves shown for the urban district support the indication of an effective use of resources to contain the spread of the virus.

PART 4

CONCLUSIONS

The methods and algorithms investigated in this dissertation provide a broad foundation for further research in the study of parameter estimation in dynamical models utilizing limited (early) data. The extension of these approaches to models incorporating additional intra- and inter-dynamics is an open avenue for future work. There are many diseases effectively modeled by SEIR-type models for which this work is applicable. Moreover, models incorporating additional reservoirs of infection (e.g., domestic poultry and mosquitoes) or additional compartments reflecting dynamics of control (e.g., vaccination) can be studied utilizing methods adapted from this research.

In Chapter 2, numerical studies using early epidemiological data were undertaken to reliably recover parameters helpful in informing responses to disease outbreaks. We have illustrated the potential of rigorous mathematical approaches for generating stable parameter estimates and useful epidemic forecasts in the context of the generalized Richards model, a simple phenomenological 4-parameter model. Our results here suggest that carefully linking mathematical models with regularization techniques could lead to improved parameter estimation and epidemic forecasts. In future work, a systematic comparison across various phenomenological models will be conducted in order to assess which model is the most effective for stable parameter estimation from early outbreak data. In an optimization algorithm, the regularization will be enforced through a special nonlinear penalty term quantifying a sub-exponential growth rate of an emerging outbreak. Further studies will make use of incidence data for Avian Influenza, Middle East Respiratory Syndrome (MERS), and Zika virus among others.

Static SEIR epidemic models that assume constant transmission rates tend to overestimate epidemic impact owing to the assumption of early exponential epidemic growth. Yet, disease transmission is not a static process, and a number of factors affect the transmission dynamics during an epidemic including the effects of reactive behavior changes, control interventions, and spatial heterogeneity that can dampen or amplify disease transmission rates. Incorporating time-dependent transmission rates in epidemic models is crucial to reliably forecast disease spread in a population. In Chapter 3, we introduce a new approach for estimating the transmission rate of an outbreak in near real time for SEIR-type epidemics in order to generate informative forecasts of epidemic impact. We show that this method is

able to provide reasonable forecasts of epidemic impact using different regularization techniques. Our methodology is designed to help with forecasting of emerging and re-emerging infectious diseases, and it could be adapted to incorporate other additional epidemiological (e.g., varying levels of infectiousness) and transmission (e.g., environmental vs. close contact transmission) mechanisms.

REFERENCES

[1] G. Chowell, "Disease outbreak data - various," 2016, School of Public Health, Georgia State University, Atlanta, Georgia.

[2] J. Hadamard, *Lectures on Cauchy's Problem: In Linear Partial Differential Equations*, ser. Mrs. Hepsa Ely Silliman memorial lectures. Yale University Press, London: Oxford University Press, 1923. [Online]. Available: https://books.google.com/books?id=vn5xmgEACAAJ

[3] M. M. Lavrent_ev, V. G. Romanov, and S. P. Shishatski_, *Ill-posed problems of mathematical physics and analysis*. American Mathematical Soc., 1986, vol. 64.

[4] C. W. Groetsch and C. Groetsch, *Inverse problems in the mathematical sciences*. Springer, 1993, vol. 52.

[5] A. Bakushinsky and A. Goncharsky, *Ill-posed problems: theory and application*. Kluver, Dordrecht, 1994.

[6] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*. Springer Science & Business Media, 1996, vol. 375.

[7] A. N. Tikhonov, A. S. Leonov, and A. Yagola, "Nonlinear ill-posed problems," in *Proceedings of the first world congress on World congress of nonlinear analysts' 92, volume I*. Walter de Gruyter & Co., 1996, pp. 505–511.

[8] V. Ivanov, "On linear ill-posed problems," in *Dokl. Akad. Nauk SSSR*, vol. 145, no. 2, 1962, pp. 270–272.

[9] V. K. Ivanov, "On ill-posed problems," *Matematicheskii sbornik*, vol. 103, no. 2, pp. 211–223, 1963.

[10] A. M. Denisov, *Elements of the theory of inverse problems*. VSP, 1999, vol. 14.

[11] V. Ivanov, V. Vasin, and V. Tanana, "Theory of ill-posed linear problems and its applications," 1978.

[12] O. Liskovets, "The method of e-quasisolutions for equations of the first kind," *Differentsial'nye Uravneniya*, vol. 9, pp. 1851–1861, 1973.

[13] A. Tikhonov, "On the ill-posed problems solution and regularization method," in *Dokl. Acad. Nauk*, vol. 151, no. 3, 1963, pp. 501–505.

[14] D. L. Phillips, "A technique for the numerical solution of certain integral equations of the first kind," *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 84–97, 1962.

[15] C. Groetsch, "The theory of tikhonov regularization for fredholm equations," *104p, Boston Pitman Publication*, 1984.

[16] B. Hofmann, "Regularization of applied inverse and ill posed problems," 1986.

[17] V. Morozov, "The principle of discrepancy in the solution of inconsistent equations by tikhonovs regularization method," *Zhurnal Vychislitelnoy matematiky i matematicheskoy fisiki*, vol. 13, no. 5, 1973.

[18] H. Engl, "Discrepancy principles for tikhonov regularization of ill-posed problems leading to optimal convergence rates," *Journal of optimization theory and applications*, vol. 52, no. 2, pp. 209–215, 1987.

[19] M. Hanke, "An ϵ-free a posteriori stopping rule for certain iterative regularization methods," *SIAM journal on numerical analysis*, vol. 30, no. 4, pp. 1208–1228, 1993.

[20] O. Scherzer, H. W. Engl, and K. Kunisch, "Optimal a posteriori parameter choice for tikhonov regularization for solving nonlinear ill-posed problems," *SIAM Journal on Numerical Analysis*, vol. 30, no. 6, pp. 1796–1838, 1993.

[21] M. Hanke and H. W. Engl, "An optimal stopping rule for the v-method for solving ill-posed problems, using christoffel functions," *Journal of Approximation Theory*, vol. 79, no. 1, pp. 89–108, 1994.

[22] H. W. Engl and A. Neubauer, "Optimal discrepancy principles for the tikh0n0v regularization of integral equations of the first kind," in *Constructive methods for the practical treatment of integral equations*. Springer, 1985, pp. 120–141.

[23] A. Neubauer, "An a posteriori parameter choice for tikhonov regularization in hilbert scales leading to optimal convergence rates," *SIAM journal on numerical analysis*, vol. 25, no. 6, pp. 1313–1326, 1988.

[24] G. Vainikko and A. Y. Veretennikov, "Iteration procedures in ill-posed problems," 1986.

[25] V. V. Vasin and A. L. Ageev, *Ill-posed problems with a priori information*. Walter de Gruyter, 1995, vol. 3.

[26] M. Hanke, "Accelerated landweber iterations for the solution of ill-posed equations," *Numerische mathematik*, vol. 60, no. 1, pp. 341–373, 1991.

[27] P. K. Lamm, "Full convergence of sequential local regularization methods for volterra inverse problems," *Inverse Problems*, vol. 21, no. 3, p. 785, 2005.

[28] F. Cakoni and D. Colton, *A qualitative approach to inverse scattering theory*. Springer, 2014.

[29] D. Colton and R. Kress, *Inverse acoustic and electromagnetic scattering theory*. Springer Science & Business Media, 2012, vol. 93.

[30] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*. Springer Science & Business Media, 2006, vol. 153.

[31] G. Wahba, "Practical approximate solutions to linear operator equations when the data are noisy," *SIAM Journal on Numerical Analysis*, vol. 14, no. 4, pp. 651–667, 1977.

[32] J. ALBER and I. RIAZANTSEVA, "Discrepancy principle in nonlinear problems with monotone discontinuous mappings-regularizing algorithm," *DOKLADY AKADEMII NAUK SSSR*, vol. 239, no. 5, pp. 1017–1020, 1978.

[33] A. G. Ramm, "Random fields estimation theory," *Mathematical and Computer Modelling*, vol. 13, no. 4, pp. 87–100, 1990.

[34] F. Liu and M. Zuhair Nashed, "Convergence of regularized solutions of nonlinear ill-posed problems with monotone operators," *Lecture Notes in Pure and Applied Mathematics*, pp. 353–361, 1996.

[35] M. Nashed and F. Liu, "On nonlinear ill-posed problems ii: Monotone operator equations and monotone variational inequalities," *Lecture Notes in Pure and Applied Mathematics*, pp. 223–240, 1996.

[36] C. R. Vogel, *Computational methods for inverse problems*. SIAM, 2002.

[37] S. Iwami, Y. Takeuchi, and X. Liu, "Avian–human influenza epidemic model," *Mathematical Biosciences*, vol. 207, no. 1, pp. 1–25, 2007.

[38] S. Iwami, Y. Takeuchi, A. Korobeinikov, and X. Liu, "Prevention of avian influenza epidemic: What policy should we choose?" *Journal of theoretical biology*, vol. 252, no. 4, pp. 732–741, 2008.

[39] S. Iwami, Y. Takeuchi, and X. Liu, "Avian flu pandemic: Can we prevent it?" *Journal of theoretical biology*, vol. 257, no. 1, pp. 181–190, 2009.

[40] K. I. Kim, Z. Lin, and L. Zhang, "Avian-human influenza epidemic model with diffusion," *Nonlinear Analysis: Real World Applications*, vol. 11, no. 1, pp. 313–322, 2010.

[41] J. Lucchetti, M. Roy, and M. Martcheva, "An avian influenza model and its fit to human avian influenza cases," *Advances in Disease Epidemiology, Nova Science Publishers, New York*, pp. 1–30, 2009.

[42] M. Martcheva and F. Hoppensteadt, "India's approach to eliminating plasmodium falciparum malaria: a modeling perspective," *Journal of Biological Systems*, vol. 18, no. 04, pp. 867–891, 2010.

[43] N. Tuncer and M. Martcheva, "Modeling seasonality in avian influenza h5n1," *Journal of Biological Systems*, vol. 21, no. 04, p. 1340004, 2013.

[44] A. Bakushinsky, "Iterative methods for nonlinear operator equations without regularity. new approach," in *Dokl. Russian Acad. Sci*, vol. 330, 1993, pp. 282–284.

[45] A. Smirnova, G. Chowell-Puente, L. deCamp, S. Moghadas, and M. J. Sheppard, "Improving epidemic size prediction through stable reconstruction of disease parameters by reduced iteratively regularized gauss–newton algorithm," *Journal of Inverse and Ill-posed Problems*, 2017.

[46] W. H. Organization *et al.*, "Statement on the 1st meeting of the ihr emergency committee on the 2014 ebola outbreak in west africa," *World Health Organization, IHR Emergency Committee regarding Ebola*, 2014.

[47] "*World Health Organization*, Ebola Situation Report - 30 March 2016," http://apps.who.int/ebola/current-situation/ebola-situation-report-30-march-2016, retrieved 7/13/15.

[48] "Frequently asked questions on ebola virus disease," World Health Organization. [Online]. Available: http://www.who.int/csr/disease/ebola/faq-ebola/en/

[49] G. Chowell, N. W. Hengartner, C. Castillo-Chavez, P. W. Fenimore, and J. Hyman, "The basic reproductive number of ebola and the effects of public health measures: the cases of congo and uganda," *Journal of theoretical biology*, vol. 229, no. 1, pp. 119–126, 2004.

[50] C. Althaus, "Estimating the reproduction number of zaire ebolavirus (ebov) during the 2014 outbreak in west africa. plos currents outbreaks. 2014."

[51] P.-F. Verhulst, "Notice sur la loi que la population suit dans son accroissement. correspondance mathématique et physique publiée par a," *Quetelet*, vol. 10, pp. 113–121, 1838.

[52] F. Richards, "A flexible growth function for empirical use," *Journal of experimental Botany*, vol. 10, no. 2, pp. 290–301, 1959.

[53] A. Tsoularis and J. Wallace, "Analysis of logistic growth models," *Mathematical biosciences*, vol. 179, no. 1, pp. 21–55, 2002.

[54] C. Viboud, L. Simonsen, and G. Chowell, "A generalized-growth model to characterize the early ascending phase of infectious disease outbreaks," *Epidemics*, vol. 15, pp. 27–37, 2016.

[55] G. Chowell, C. Viboud, J. M. Hyman, and L. Simonsen, "The western africa ebola virus disease epidemic exhibits both global exponential and local polynomial growth rates," *arXiv preprint arXiv:1411.7364*, 2014.

[56] G. Chowell, C. Viboud, L. Simonsen, and S. M. Moghadas, "Characterizing the reproduction number of epidemics with early subexponential growth dynamics," *Journal of The Royal Society Interface*, vol. 13, no. 123, p. 20160659, 2016.

[57] B. Szendroi and G. Csányi, "Polynomial epidemics and clustering in contact networks," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 271, no. Suppl 5, pp. S364–S366, 2004.

[58] M. E. Turner, E. L. Bradley, K. A. Kirk, and K. M. Pruitt, "A theory of growth," *Mathematical Biosciences*, vol. 29, no. 3-4, pp. 367–373, 1976.

[59] F. Cavallini, "Fitting a logistic curve to data," *The College Mathematics Journal*, vol. 24, no. 3, 1993.

[60] M. Works, *R2016b*. Natick, Massachusetts: The MathWorks Inc., 2016.

[61] A. Bakushinsky and M. Y. Kokurin, "Iterative methods for ill-posed operator equations with smooth operators," 2004.

[62] F. Bauer, T. Hohage, and A. Munk, "Iteratively regularized gauss–newton method for nonlinear inverse problems with random noise," *SIAM Journal on Numerical Analysis*, vol. 47, no. 3, pp. 1827–1846, 2009.

[63] B. Kaltenbacher, A. Neubauer, and O. Scherzer, *Iterative regularization methods for nonlinear ill-posed problems*. Walter de Gruyter, 2008, vol. 6.

[64] M. Y. Kokurin and A. B. Bakushinsky, "Iteratively regularized gauss-newton methods under random noise," in *Inverse Problems and Applications*. Springer, 2015, pp. 1–11.

[65] S. Langer and T. Hohage, "Convergence analysis of an inexact iteratively regularized gauss-newton method under general source conditions," *Journal of Inverse and Ill-posed Problems jiip*, vol. 15, no. 3, pp. 311–327, 2007.

[66] A. Smirnova, "On convergence rates for iteratively regularized procedures with linear penalty terms," *Inverse Problems*, vol. 28, no. 8, p. 085005, 2012.

[67] A. N. Tikhonov, A. Goncharsky, V. Stepanov, and A. G. Yagola, *Numerical methods for the solution of ill-posed problems*. Springer Science & Business Media, 2013, vol. 328.

[68] A. G. Yagola, "Ill-posed problems and methods for their numerical solution," in *Optimization and Regularization for Computational Inverse Problems and Applications*. Springer, 2010, pp. 17–34.

[69] S. Wright and J. Nocedal, "Numerical optimization," *Springer Science*, vol. 35, pp. 67–68, 1999.

[70] A. Smirnova, R. A. Renaut, and T. Khan, "Convergence and application of a modified iteratively regularized gauss–newton algorithm," *Inverse Problems*, vol. 23, no. 4, p. 1547, 2007.

[71] Q. Jin, "Further convergence results on the general iteratively regularized gauss-newton methods under the discrepancy principle," *Mathematics of Computation*, vol. 82, no. 283, pp. 1647–1665, 2013.

[72] M. Lipsitch, T. Cohen, B. Cooper, J. M. Robins, S. Ma, L. James, G. Gopalakrishna, S. K. Chew, C. C. Tan, M. H. Samore *et al.*, "Transmission dynamics and control of severe acute respiratory syndrome," *Science*, vol. 300, no. 5627, pp. 1966–1970, 2003.

[73] C. M. Rivers, E. T. Lofgren, M. Marathe, S. Eubank, and B. L. Lewis, "Modeling the impact of interventions on an epidemic of ebola in sierra leone and liberia," *arXiv preprint arXiv:1409.4607*, 2014.

[74] M. Pollicott, H. Wang, and H. Weiss, "Extracting the time-dependent transmission rate from infection data via solution of an inverse ode problem," *Journal of biological dynamics*, vol. 6, no. 2, pp. 509–523, 2012.

[75] A. Lange, "Reconstruction of disease transmission rates: applications to measles, dengue, and influenza," *Journal of theoretical biology*, vol. 400, pp. 138–153, 2016.

[76] P. E. Lekone and B. F. Finkenstädt, "Statistical inference in a stochastic epidemic seir model with control intervention: Ebola as a case study," *Biometrics*, vol. 62, no. 4, pp. 1170–1177, 2006.

[77] O. N. Bjørnstad, B. F. Finkenstädt, and B. T. Grenfell, "Dynamics of measles epidemics: estimating scaling of transmission rates using a time series sir model," *Ecological Monographs*, vol. 72, no. 2, pp. 169–184, 2002.

[78] B. P. Taylor, J. Dushoff, and J. S. Weitz, "Stochasticity and the limits to confidence when estimating r0 of ebola and other emerging infectious diseases," *Journal of Theoretical Biology*, vol. 408, pp. 145–154, 2016.

[79] J. M. Ponciano and M. A. Capistrán, "First principles modeling of nonlinear incidence rates in seasonal epidemics," *PLoS Comput Biol*, vol. 7, no. 2, p. e1001079, 2011.

[80] M. E. Newman, "Spread of epidemic disease on networks," *Physical review E*, vol. 66, no. 1, p. 016128, 2002.

[81] M. A. Kiskowski, "A three-scale network model for the early growth dynamics of 2014 west africa ebola epidemic," *PLOS Currents Outbreaks*, 2014.

[82] Y. Xia, O. N. Bjørnstad, and B. T. Grenfell, "Measles metapopulation dynamics: a gravity model for epidemiological coupling and dynamics," *The American Naturalist*, vol. 164, no. 2, pp. 267–281, 2004.

[83] S. Cauchemez, A.-J. Valleron, P.-Y. Boëlle, A. Flahault, and N. M. Ferguson, "Estimating the impact of school closure on influenza transmission from sentinel data," *Nature*, vol. 452, no. 7188, pp. 750–754, 2008.

[84] E. K. Szusz, L. P. Garrison, and C. T. Bauch, "A review of data needed to parameterize a dynamic model of measles in developing countries," *BMC research notes*, vol. 3, no. 1, p. 75, 2010.

[85] B. F. Finkenstädt and B. T. Grenfell, "Time series modelling of childhood diseases: a dynamical systems approach," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 49, no. 2, pp. 187–205, 2000.

[86] M. A. Capistrán, M. A. Moreles, and B. Lara, "Parameter estimation of some epidemic models. the case of recurrent epidemics caused by respiratory syncytial virus," *Bulletin of mathematical biology*, vol. 71, no. 8, pp. 1890–1901, 2009.

[87] K. Hadeler, "Parameter identification in epidemic models," *Mathematical biosciences*, vol. 229, no. 2, pp. 185–189, 2011.

[88] A. Bakushinsky, A. Smirnova, and H. Liu, "A nonstandard approximation of pseudoinverse and a new stopping criterion for iterative regularization," *Journal of Inverse and Ill-posed Problems*, vol. 23, no. 3, pp. 195–210, 2015.

[89] V. A. Morozov, *Methods for solving incorrectly posed problems*. Springer Science & Business Media, 2012.

Appendix A

MATLAB CODES

A.1 MATLAB CODE 1 - RIRGN

```
function gRichards_RIRGN_DIS
% reduced (completely) Jacobian; simplified formulas
close all
clear all
clc
format long
warning('off','all')
%-------------------------------------------------------------------------
% DATA SELECTION AND CERTAIN ALGORITHM PARAMETER SET
%-------------------------------------------------------------------------
DSet = str2double(input([...
    'Choose from among the following data sets by number indicated',...
        '\n 1) Sierra Leone',...
        '\n 2) Liberia',...
        '\n 3) Guinea',...
        '\n',...
        '\n Which?  '],'s'));
   while isnan(DSet) || fix(DSet) ~= DSet || DSet<1 ||DSet>3
       DSet = str2double(input(...
           'Please enter and INTEGER between 1 and 3: ', 's'));
   end
   switch DSet
       case 1
           load SLcumcases.txt;
           tdata_full = SLcumcases(:,1);
           Cdata_full = SLcumcases(:,2);
```

```
        m = 5;
        lambda0 = 5e-4;
        tau_actual = 28; %SL
        PDWk1 = 23; PDWk2 = 28;
        y1f1 = [0,700];
        y1f2 = [0,25000];
        y1f3 = [0,65]; % SL
        RegID = 'Sierra Leone';
    case 2
        load LIBcumcases_rev.txt;
        tdata_full = LIBcumcases_rev(:,1);
        Cdata_full = LIBcumcases_rev(:,2);
        m = 5;
        lambda0 = 5e-4;
        tau_actual = 18; %L
        PDWk1 = 11; PDWk2 = 18;
        y1f1 = [0,800];
        y1f2 = [0,18000]; % Liberia
        y1f3 = [0,45];
        RegID = 'Liberia';
    case 3
        load GUcumcases.txt;
        tdata_full = GUcumcases(:,1);
        Cdata_full = GUcumcases(:,2);
        m = 5;
        lambda0 = 1e-3;
        RegID = 'Guinea';
        tau_actual = 48; %G
        PDWk1 = 34; PDWk2 = 48
        y1f1 = [0,250];
        y1f2 = [0,7000];
        y1f3 = [0,90];
end
```

```
global Cdata Cdata_inc K KINV r j
Cdata_inc_full = [Cdata_full(1,1); diff(Cdata_full(:,1))];
mub = length(tdata_full);
sigma = 1e-4; % Line search parameter
%-------------------------------------------------------------------------
% DETERMENISTIC RECOVERY - ODE23S AND IRGN
%-------------------------------------------------------------------------

RD = zeros(mub,1);
KD = zeros(mub,1);
rD = zeros(mub,1);
kj = zeros(mub,4);
CIpar = zeros(mub,8);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameters of the Numerical Algorithm
NUM_IT = 20; % Maximum number of Gauss-Newton iterations - partial data
w = 0.5;  % lambda = lambda0*exp(-w*n), variable regularization parameter;
k0 = [.5 10.11 1 1]';
k = k0;

for j = m:mub
    tdata = tdata_full(1:j,1);
    Cdata_inc = Cdata_inc_full(1:j,1);
    Cdata = Cdata_full(1:j,1);
    k = [.5 min(10.11,k(2)) 1 1]';
    xi = [1 60 1.5 1]';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Printing Results

disp('ITERATIVE RESULTS - GENERALIZED RICHARDS')
disp('
    ----------------------------------------------------------------------------
```

```
    ')
disp('Iter ... Line ... Discrepancy ... Cond(F_prime) ... Cond(F_prm_reg) ...
    Iter lambda ... Iter alpha')
disp('
    ----------------------------------------------------------------------------
    ')

for count = 1:NUM_IT

[HPRIME,H] = model_2(k,tdata);
RDS = norm(KINV*Cdata_inc-HPRIME)/norm(KINV*Cdata_inc);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STOP IF CONVERGENCE OR DIVERGENCE DETECTED
    if (RDS < 1e-10)
        fprintf('Convergence Detected!\n');
        break;
    else
        if (RDS > 100000)
            fprintf('Divergence Detected!\n');
            break;
        end;
    end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Impose nonnegativity
if k(1)<0
    k(1)=.1;
end
for i = 3:4
    if k(i)<0
        k(i)=.5;
    end
end
```

```
if k(2)<tdata(1)
    k(2)=tdata(1);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assemble Algorithmic Scheme
HP = model_2d(k,tdata);

HPR = zeros(j,5);

c = k(3)+k(4);
Ha = c*HP(:,1).^(c-1);
Hp = k(3)*HP(:,1).^(k(3)-1);

HPR(:,1) = k(1)*HP(:,1).^k(3).*(1-HP(:,1).^k(4));
HPR(:,2) = HP(:,1).^k(3).*(1-HP(:,1).^k(4)) + k(1)*(Hp - Ha).*HP(:,2);
HPR(:,3) = k(1)*(Hp - Ha).*HP(:,3);
HPR(:,4) = k(1)*(HP(:,1).^k(3).*(1-HP(:,1).^k(4)).*log(HP(:,1)) + (Hp - Ha).*
    HP(:,4));
HPR(:,5) = k(1)*((Hp - Ha).*HP(:,5) - H.^(k(3)+k(4)).*log(HP(:,1)));

FPFP = HPR(:,2:5)'*HPR(:,2:5);
FP = HPR(:,2:5);
W = diag([100 1 100 100]);
lambda = lambda0*exp(-w*count);    % Regularization parameter




% Update the iterative solution EQN (2.24)
step = -(FPFP + lambda*W)\(FP'*(HPRIME-KINV*Cdata_inc) + lambda*W*(k-xi));
Jk = FP'*(HPRIME-KINV*Cdata_inc);
D2k = (HPRIME-KINV*Cdata_inc)'*(HPRIME-KINV*Cdata_inc);
SPk = Jk'*step;

% Start the line search
```

```
alpha = 1;
for line = 1:5
    alpha = alpha/2;
    k1 = real(k + alpha*step);
    if k1(1)<0
        k1(1)=.1;
    end
    for i = 3:4
        if k1(i)<0
            k1(i)=.5;
        end
    end
    if k1(2)<tdata(1)
        k1(2)=tdata(1);
    end
    [HPRIME,~] = model_2(k1,tdata);
    D2k1 = (HPRIME-KINV*Cdata_inc)'*(HPRIME-KINV*Cdata_inc);
    if D2k1 < sigma*alpha*SPk + D2k
        %fprintf('Line Search Success!\n');
        break;
    end
end  % End of the line search inner loop


RD_Temp = norm(KINV*Cdata_inc-HPRIME)/norm(KINV*Cdata_inc);
if (1.0*RDS < RD_Temp)
    RD(j) = RDS;
        %fprintf('Stopping Time!\n');
        break;
end
k = k1;
% Compute condition number for PP'*PP
condFP = cond(FPFP);
condFP_reg = cond(FPFP + lambda*W);
```

```
% end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Creation of matrix for output of TABLE VALUES
fprintf('%2.1f ... %2.1f ... %8.6e  ... %10.6e ... %10.6e ... %8.6e ... %8.6e\n
    ',...
    count,line,RD_Temp, condFP,condFP_reg, lambda, alpha);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end % End of the iterative outer loop
fprintf('n = %2.1f, tau =%16.5f\n', j, k(2));
disp('
    ==========================================================================
    ')

%CREATION OF MATRIX FOR OUTPUT OF TABLE VALUES
RD(j) = RD_Temp;
kj(j,:) = k(:);
KD(j) = K;
rD(j) = r;
% Computation of confidence intervals
residual = real(K*HPRIME - Cdata_inc);
FPC = real(K*FP);

ci = nlparci(k,residual,'jacobian',FPC);
CIpar(j,1) = ci(1,1); % b lower bounds of confidence intervals
CIpar(j,2) = ci(1,2); % b upper bounds of confidence intervals
CIpar(j,3) = ci(2,1); % tau lower bounds of confidence intervals
CIpar(j,4) = ci(2,2); % tau upper bounds of confdience intervals
CIpar(j,5) = ci(3,1); % p lower bounds of confidence intervals
CIpar(j,6) = ci(3,2); % p upper bounds of confidence intervals
CIpar(j,7) = ci(4,1); % a lower bounds of confidence intervals
CIpar(j,8) = ci(4,2); % a upper bounds of confdience intervals
```

```
taulow = kj(m:mub,2) - CIpar(m:mub,3); % Correct lower bounds for error bar
    plot
tauup = CIpar(m:mub,4) - kj(m:mub,2); % Correct upper bounds for error bar
    plot

end

disp('RECOVERED PARAMETERS FROM IRGN METHOD - GENERALIZED RICHARDS')
disp('
    ----------------------------------------------------------------------------------------------------
    ')
disp('j ..... b ..... tau ..... p ..... a ..... r
    ..... K ..... RD ')
disp('
    ----------------------------------------------------------------------------------------------------
    ')

for j = m:mub
fprintf('%2.1f ... %8.5f ... %8.5f ... %7.5f ... %7.5f ... %11.9f ... %9.2f ...
    %7.5f \n',...
    j, kj(j,1), kj(j,2), kj(j,3), kj(j,4), rD(j), KD(j), RD(j));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x1 = [tau_actual, tau_actual];

figure1 = figure;
axes2 = axes('Parent',figure1,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
```

```
[~,~] = model_2(kj(PDWk1,:),tdata);
[T1,Y1] = model_p(kj(PDWk1,:),tdata, K);
[~,~] = model_2(kj(PDWk2,:),tdata);
[T2,Y2] = model_p(kj(PDWk2,:),tdata, K);
[~,~] = model_2(kj(j,:),tdata);
[T3,Y3] = model_p(k,tdata,K);
plot(tdata,Cdata_inc_full, '*b', T1, Y1(:,1),'-k', T2, Y2(:,1),'-g',T3,Y3(:,1)
    ,'-r','linewidth',2.5);
plot(x1,y1f1,'--b')
legend({'Incidence Cases', sprintf('%d Weeks of Data',PDWk1), sprintf('%d
    Weeks of Data',PDWk2),'Full Data'},...
        'FontSize',14,'Location','northeast');
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',14,'FontName','Arial');
ylabel({'Number of Incidence Cases'},'LineWidth',2,'FontSize',14,'FontName','
    Computer Modern');
%title(sprintf('Recovered Incidence Curves - %s',RegID),'FontSize',12)
figure(figure1)

figure2 = figure;
axes2 = axes('Parent',figure2,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
xlim([m-1,mub+1]);
plot(tdata(m:mub,1), KD(m:mub,1), '-r', 'linewidth',2.5)
x1 = [tau_actual,tau_actual];
plot(x1,y1f2,'--b')
legend({'Capacity of the Outbreak'},...
    'FontSize',14,'Location','northeast')
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',14,'FontName','Arial');
ylabel({'Values of K'},'LineWidth',2,'FontSize',14,'FontName','Computer Modern
    ');
```

```
%title(sprintf('Recovered K, capacity, with Confidence Intervals - %s', RegID
    ),'FontSize',12)
figure(figure2)

figure3 = figure;
axes2 = axes('Parent',figure3,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
axis([m-1,mub+1,0,mub-1]);
hold(axes2,'all');
errorbar(tdata(m:mub,1),kj(m:mub,2),taulow,tauup,'k')
plot(tdata(m:mub,1), kj(m:mub,2), '-m', 'linewidth',2.5)
x1 = [tau_actual,tau_actual];
x2 = [m-1,mub+1]; y2 = [tau_actual,tau_actual];
plot(x1,y1f3,'--b',x2,y2,'--b')
legend({'CIs of Computed Turning Point','Computed Turning Point','Actual
    Turning Point'},...
    'FontSize',14,'Location','northeast')
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',14,'FontName','Arial');
ylabel({'Values of \tau'},'LineWidth',2,'FontSize',14,'FontName','Computer
    Modern');
%title(sprintf('Recovered \\tau, turning point, with Confidence Intervals - %
    s', RegID),'FontSize',12)
figure(figure3)

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function dh = model_1(~,h,k)  %EQN 2.10

dh(1) = k(1)*h(1).^k(3)*(1-h(1).^k(4));

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function dh = model_1d(~,h,k)  %Partials of H

dh = zeros(5,1);

c = k(3)+k(4);
ha = c*h(1).^(c-1);
hp = k(3)*h(1).^(k(3)-1);

dh(1) = k(1)*h(1).^k(3)*(1-h(1).^k(4));
dh(2) = h(1).^k(3)*(1-h(1).^k(4)) + k(1)*(hp - ha).*h(2);
dh(3) = k(1)*(hp - ha).*h(3);
dh(4) = k(1)*(h(1).^k(3)*(1-h(1).^k(4)).*log(h(1)) + (hp -ha).*h(4));
dh(5) = k(1)*((hp - ha).*h(5) - h(1).^(k(3)+k(4)).*log(h(1)));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [HPRIME,H] = model_2(k,tdata)   %Solve BVP (2.10) by cases

global Cdata Cdata_inc KC K KINV r j
tdata_tau = zeros(j+1,1);
H = zeros(j,1);
options = odeset('RelTol',1e-6,'AbsTol',1e-8); % Sierra Leone
% options = odeset('RelTol',1e-5,'AbsTol',1e-7); % Liberia

if k(2) >= tdata(1) && k(2) < tdata(j)
for n = 1:j-1
    if  k(2) == tdata(n)
        k(2) = k(2) + .001;
    end
    if k(2) > tdata(n) && k(2) < tdata(n+1)
        N = n;
        tdata_tau(1:n) = tdata(1:n);
```

```
        tdata_tau(n+1) = k(2);
        tdata_tau(n+2:j+1) = tdata(n+1:j);
    end
end

[~,HF] = ode23s(@(t,h) (model_1(t,h,k)),tdata_tau(N+1:j+1,1),...
    (k(3)/(k(3)+k(4)))^(1/k(4)),options);
[~,HB] = ode23s(@(t,h) (model_1(t,h,k)),tdata_tau(N+1:-1:1,1),...
    (k(3)/(k(3)+k(4)))^(1/k(4)),options);

H = vertcat(flipud(HB(1:N)),HF(2:j+1-N));
end

if k(2) >= tdata(j)
    sol = ode23s(@(t,h) (model_1(t,h,k)),[k(2) tdata(1)],...
        (k(3)/(k(3)+k(4)))^(1/k(4)),options);
    H = deval(sol,tdata)';
end

KC = (H'*Cdata)/(H'*H);
if KC > 1e6
    KC = 1e6;
end
if KC < Cdata(j,1)
    KC = Cdata(j,1);
end

HPRIME = k(1)*H.^k(3).*(1-H.^k(4));
K = (HPRIME'*Cdata_inc)/(HPRIME'*HPRIME);
if K > 1e6
    K = 1e6;
end
if K < Cdata(j,1)
    K = Cdata(j,1);
```

```
end

r = k(1)*K^(1-k(3));
KINV = 1/K;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function HPARTIAL = model_2d(k,tdata) % Solve System for Partials of H

global j
tdata_tau = zeros(j+1,1);
HPARTIAL = zeros(j,5);
options = odeset('RelTol',1e-6,'AbsTol',1e-8);

initial = [(k(3)/(k(3)+k(4)))^(1/k(4)), ...
    0, ...
    -k(1)*(k(3)/(k(3)+k(4)))^(k(3)/k(4))*k(4)/(k(3)+k(4)), ...
    k(3)^(1/k(4)-1)/(k(3)+k(4))^(1/k(4)+1),...
    -(k(3)/(k(3)+k(4)))^(1/k(4))*((k(3)+k(4))*...
    log(k(3)/(k(3)+k(4)))+k(4))/((k(3)+k(4))*k(4)^2)]';

if k(2) >= tdata(1) && k(2) < tdata(j)
for n = 1:j-1
    if  k(2) == tdata(n)
        k(2) = k(2) + .001;
    end
    if k(2) > tdata(n) && k(2) < tdata(n+1)
        N = n;
        tdata_tau(1:n) = tdata(1:n);
        tdata_tau(n+1) = k(2);
        tdata_tau(n+2:j+1) = tdata(n+1:j);
    end
end
```

```
[~,HF] = ode23s(@(t,h) (model_1d(t,h,k)),tdata_tau(N+1:j+1,1)',...
    initial, options);
[~,HB] = ode23s(@(t,h) (model_1d(t,h,k)),tdata_tau(N+1:-1:1,1)',...
    initial, options);

for i=1:5
HPARTIAL(:,i) = vertcat(flipud(HB(1:N,i)),HF(2:j+1-N,i));
end
% HPARTIAL
end

if k(2) >= tdata(j)
    sol = ode23s(@(t,h) (model_1d(t,h,k)),[k(2) tdata(1)],...
        initial, options);
    HPARTIAL = deval(sol,tdata)';
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [T,Y] = model_p(k,tdata,Kp) %Reconstruct Curves

global j
tdata_tau = zeros(j+1,1);
H = zeros(j,1);
options = odeset('RelTol',1e-6,'AbsTol',1e-8);

if k(2) >= tdata(1) && k(2) < tdata(j)
for n = 1:j-1
    if k(2) == tdata(n)
        k(2) = k(2) + .001;
    end
    if k(2) > tdata(n) && k(2) < tdata(n+1)
```

```
        N = n;
        tdata_tau(1:n) = tdata(1:n);
        tdata_tau(n+1) = k(2);
        tdata_tau(n+2:j+1) = tdata(n+1:j);
    end
end


[TF,HF] = ode23s(@(t,h) (model_1(t,h,k)),[tdata_tau(N+1) tdata_tau(j+1)],...
    (k(3)/(k(3)+k(4)))^(1/k(4)),options);
[TB,HB] = ode23s(@(t,h) (model_1(t,h,k)),[tdata_tau(N+1) tdata_tau(1)],...
    (k(3)/(k(3)+k(4)))^(1/k(4)),options);

H = vertcat(flipud(HB),HF);
T = vertcat(flipud(TB),TF);
end
if k(2) >= tdata(j)
    [T,H] = ode23s(@(t,h) (model_1(t,h,k)),[k(2) tdata(1)],...
        (k(3)/(k(3)+k(4)))^(1/k(4)),options);
end

Y = Kp*k(1)*H.^k(3).*(1-H.^k(4));
end
```

A.2 MATLAB CODE 2 - Uncertainty in the Reconstruction of $\beta(t)$ - MTSVD

```
function beta_line_MTSVD_uncertainty_Sim_CI_OPT_A_DP2
% This function uses incidence data to create confidence interval plots for
% time-dependent data.  Curves and time-dependent data files can be created
% for use in other plotting programs.  Additional data sets may be added.
close all
clear all
clc
```

```
format long
warning('off','all')

global N kappa gamma j C0 E0 I0 S0 mub
%-------------------------------------------------------------------------
% USER INPUT
FILES = input('Do you wish to create csv files for data generated? (y/n)  ','s
    ');
if strcmpi(FILES,'y')
    Curve_filename = input('File Name for Poisson Curves ','s');
    Curve_filename_full =strcat(Curve_filename,'.csv');
    Beta_1m_filename = input('File Name for 1m Beta in Reconstruction ','s');
    Beta_1m_filename_full = strcat(Beta_1m_filename,'.csv');
    Beta_filename = input('File Name for t Beta in Reconstruction ','s');
    Beta_filename_full = strcat(Beta_filename,'.csv');
else
    Curve_filename_full =strcat('x.csv');
    Beta_1m_filename_full = strcat('y.csv');
    Beta_filename_full = strcat('z.csv');
end
DSet = str2double(input(['Choose from among the following data sets by number
    indicated',...
        '\n 1) Sim 1',...
        '\n 2) Sierra Leone',...
        '\n',...
        '\n Which?  '],'s'));
while isnan(DSet) || fix(DSet) ~= DSet || DSet<1 ||DSet>2
    DSet = str2double(input('Please enter and INTEGER between 1 and 2: ', 's')
        );
end

NumCurves = str2double(input('How many curves? ','s'))
while isnan(NumCurves) || fix(NumCurves) ~= NumCurves || NumCurves<1
    NumCurves = str2double(input('Please enter a positive INTEGER: ', 's'));
```

```
end
%-------------------------------------------------------------------------
% END User Input

switch DSet
    case 1
        DATA = csvread('Sim_1_VB.csv');%load SLcumcases.txt;
        tdata_full = DATA(:,1);
        Cdata_full = DATA(:,2);
        N = 1.5e6;
        lambda = 3.65e-04;
        RegID = 'Simulated Data';
    case 2
        load SLcumcases.txt;
        tdata_full = SLcumcases(:,1);
        Cdata_full = SLcumcases(:,2);
        Cdata_full = [Cdata_full(1,1);diff(Cdata_full)];
        N = 6e6;
        lambda = 2.25e-05;
        RegID = 'Sierra Leone';
end
%-------------------------------------------------------------------------
% Set program values
Cdata_inc_full = Cdata_full;
Cdata_full = cumsum(Cdata_inc_full);
kappa = 7/8;
gamma = 7/6;
S0 = N;
C0 = Cdata_full(1,1);
E0 = Cdata_inc_full(1,1)/kappa;
I0 = C0;
mub = length(tdata_full);
m = 6;
j = mub;
```

```
n = ceil(2*mub);
tdata = tdata_full(1:mub,1);
%-------------------------------------------------------------------------
% Generate Poisson Curves
nmb = 0;
yi = Cdata_inc_full;
curves = [];
for iter = 1:NumCurves
    nmb = nmb + 1
    yirData = zeros(length(yi),1);
    yirData(1) = yi(1);
    for t = 2:length(yi)
        tau = abs(yi(t));
        yirData(t,1) = poissrnd(tau,1,1);
    end
    curves = [curves (yirData)];
end
csvwrite(Curve_filename_full,curves);
%-------------------------------------------------------------------------
% Plot Figure for Poisson Curves
figure5 = figure;
axes2 = axes('Parent',figure5,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
       0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
plot(tdata_full,curves,'c');
h1 = plot(tdata_full,curves(1:mub,1),'c');
h2 = plot(tdata_full,mean(curves,2),'-k', 'LineWidth',2);
h3 = plot(tdata_full,Cdata_inc_full,'*k', 'LineWidth',2);
legend([h1 h2 h3], {'Poisson Noise', 'Mean Value', 'Simulated Data'},...
   'FontSize',12,'Location','best') % SL
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','Arial');
ylabel({'\beta(t)'},'LineWidth',2,'FontSize',12,'FontName','Computer Modern');
```

```
title(sprintf('Incidence Data Generated by Recovered \\beta - %s', RegID),'
    FontSize',12)
figure(figure5)
%-------------------------------------------------------------------------
% Set vectors and counters for iterations
nmb = 0;
timevect = linspace(tdata_full(1,1), tdata_full(mub,1), 1000);
BetaExp = zeros(1000, NumCurves);
BetaExpCI = zeros(length(tdata_full),NumCurves);
%-------------------------------------------------------------------------
% Iterate the algorithm by Number of Curves Selected
for iter = 1:NumCurves
    nmb = nmb + 1
    ExpIncData =curves(1:mub,iter);
    %  Construct A
    A = zeros(j-1,n+1);
    [f,K] = kernel(ExpIncData, cumsum(ExpIncData), tdata_full);
    K_sp = spline(tdata, K);
    a = tdata_full(1); b = tdata_full(mub);
    for k1 = 1:n+1
        for k2 = 1:j-1
            A(k2,k1) = integral(@(x) leg(x, k1-1, a, b).*ppval(K_sp,x),
                tdata_full(1), tdata_full(k2+1));
        end
    end
    % End Construct A
condition = cond(A);  % Expose for display if desired
%-------------------------------------------------------------------------
% Subroutine for Regularization Parameter Selection MTSVD
[lambda RD] = RelDisc(A,f,lambda,Cdata_inc_full,n,tdata_full);
%-------------------------------------------------------------------------
% Apply Regularization Parameter and Solve MTSVD
[U,S,V] = svd(A);
s = diag(S);
```

```
PIS = zeros(j-1,n+1);
for i = 1:j-1
    if s(i) >= lambda
        PIS(i,i) = 1/s(i);
    else
        PIS(i,i) = 1/lambda;
    end
end
PIA = V*(PIS)'*U';
Coef = PIA*f(2:j,1);
%-------------------------------------------------------------------------
% Produce output vectors
ybeta = approx(timevect,n,Coef,tdata_full(1),tdata_full(mub));
ybetaCI = approx(tdata,n,Coef,tdata_full(1),tdata_full(mub));
BetaExp(:, iter) = ybeta;
BetaExpCI(:, iter) = ybetaCI;
end
% End Iterations for Number of Curves

%-------------------------------------------------------------------------
% Write Data Files
csvwrite(Beta_1m_filename_full,BetaExp);
csvwrite(Beta_filename_full,BetaExpCI);

%-------------------------------------------------------------------------
% Figure Beta with all reconstructions
figure6 = figure;
axes2 = axes('Parent',figure6,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
plot(timevect,BetaExp,'g', 'LineWidth',1)
h1 = plot(timevect,BetaExp(1:1000,1),'g', 'LineWidth',1);
```

```
h2 = plot(timevect,mean(BetaExp,2),'-k', 'LineWidth',2);
legend([h1 h2],{'Uncertainty in \beta(t)', 'Mean Value'},...
    'FontSize',12,'Location','best') % SL
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','Computer
    Modern');
ylabel({'\beta(t)'},'LineWidth',2,'FontSize',12,'FontName','Computer Modern');
title(sprintf('Uncertainty in the Reconstruction of \\beta(t) - %s', RegID),'
    FontSize',12)
axis([tdata_full(1) tdata_full(end) 0 2])
figure(figure6)
%-------------------------------------------------------------------------
% Prepare and Plot Beta with Confidence Intervals
for i = tdata_full(1):tdata_full(j)
    pd=fitdist(BetaExpCI(i,1:NumCurves)','Normal');
    mu(i) = pd.mu;
    sig(i) = pd.sigma;
end
figure7 = figure;
axes2 = axes('Parent',figure7,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
h4 = errorbar(tdata_full,mu(tdata_full),1.96*sig(tdata_full),'r');
h3 = plot(tdata,mu(tdata_full),'k','LineWidth',2);
legend([h3 h4],{'Mean Value \beta(t)','Confidence Intervals'},...
    'FontSize',12,'Location','best') % SL
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','Computer
    Modern');
ylabel({'\beta(t)'},'LineWidth',2,'FontSize',12,'FontName','Computer Modern');
title(sprintf('Uncertainty in the Reconstruction of \\beta(t) - %s', RegID),'
    FontSize',12)
axis([tdata_full(1) tdata_full(end) 0 2])
figure(figure7)
```

```
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [f,K] = kernel(Cdata_inc, Cdata, tdata)
global N kappa gamma j C0 E0 I0 S0
K = zeros(j,1);
f = zeros(j,1);
S_p = spline(tdata(1:j,1),Cdata_inc(1:j,1));
It = zeros(j,1);
for i = 1:j
      It(i,1) = integral(@(t) exp(-gamma*(tdata(i,1) - t)).*ppval(S_p,t),
          tdata(1), tdata(i));
      K(i,1) = I0*exp(-gamma*(tdata(i,1) - tdata(1,1))) + It(i,1);
      f(i,1) = - log((- Cdata_inc(i,1)/kappa - Cdata(i,1)+ E0 + C0)/S0 + 1);
end
K = K/N;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    function P = leg(x, k, a, b)
        t = (2.*x - a - b)./(b - a);
                if k == 0
                    P1 = 1; P = P1;
                elseif k == 1
                     P2 = t; P = P2;
                else
                        P1 = 1; P2 = t;
                        for i = 2:k
```

```
                P3 = ((2*(i-1)+1).*t.*P2 - (i-1).*P1)./i;
                P1 = P2; P2 = P3;
                    end
                    P = P3;
            end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = approx(x,m,c,a,b)
        z = 0;
            for k = 1:m+1
            z = z + c(k).*leg(x, k-1, a, b);
            end
            if z < 0
                z = 0;
            end
        end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function dy = sstm(x,y,Coef,tdata_full,number)
global N kappa gamma mub

dy = zeros(3,1);
dy(1) = -approx(x,number,Coef,tdata_full(1),tdata_full(mub)).*y(1).*y(3)/N;
dy(2) = approx(x,number,Coef,tdata_full(1),tdata_full(mub)).*y(1).*y(3)/N -
    kappa*y(2);
dy(3) = kappa*y(2) - gamma*y(3);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [T,F] = operator(tdata_full,Coef,number)
```

```
global S0 E0 I0 kappa mub

options = odeset('RelTol',1e-4,'AbsTol',1e-6);
[T,Y] = ode23s(@(x,y) sstm(x,y,Coef,tdata_full,number),tdata_full(1:mub,1),[S0
    E0 I0], options);

F = kappa*Y(:,2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [lam RD] = RelDisc(A,f,temp,Cdata_inc,n,tdata_full)
global j
Lmcount = 1;
OP = [];
%MTSVD
[U,S,V] = svd(A);
s = diag(S);
% Parameter Selection by broad range
avec = linspace(temp/10,temp*10,40);
for pw = 1:length(avec)
    lambda = avec(pw);

    PIS = zeros(j-1,n+1);
    for i = 1:j-1
        if s(i) >= lambda
            PIS(i,i) = 1/s(i);
        else
            PIS(i,i) = 1/lambda;
        end
    end
    PIA = V*(PIS)'*U';
    Coef = PIA*f(2:j,1);
    [Tt, Ft] = operator(tdata_full,Coef,n);
```

```
    RD = norm(Ft(1:j)-Cdata_inc)/norm(Cdata_inc);
    OP(Lmcount,1:2) = [RD lambda];
    Lmcount = Lmcount + 1;
end
OP
[Val Ix] = min(OP(:,1));
% Narrow the Range
if Ix>1
    if Ix==40
        avec = linspace(OP(end-1,2),OP(end,2),20);
    else
        avec = linspace(OP(Ix-1,2),OP(Ix+1,2),20);
    end
else
    avec = linspace(OP(1,2),OP(3,2),20);
end
% Parameter Selection by narrow range
OP=[];
Lmcount = 1;
for pw = 1:length(avec)
    lambda = avec(pw);
    PIS = zeros(j-1,n+1);
    for i = 1:j-1
        if s(i) >= lambda
            PIS(i,i) = 1/s(i);
        else
            PIS(i,i) = 1/lambda;
        end
    end
    PIA = V*(PIS)'*U';
    Coef = PIA*f(2:j,1);
    [Tt, Ft] = operator(tdata_full,Coef,n);
    RD = norm(Ft(1:j)-Cdata_inc)/norm(Cdata_inc);
    OP(Lmcount,1:2) = [RD lambda];
```

```
    Lmcount = Lmcount + 1;
end
OP
%-------------------------------------------------------------------------
% Figure Plot Parameter/RD - COMMENT OUT WITH LARGE NUMBER OF ITERATIONS
figure9 = figure;
axes2 = axes('Parent',figure9,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
LAM = OP(:,2);
RDis = OP(:,1);
plot(LAM,RDis,'b','LineWidth',2)
legend({sprintf('MTSVD')})%,'Interpreter','latex')
xlabel({'\alpha'},'LineWidth',2,'FontSize',12,'FontName','Computer Modern');
ylabel({'Relative Discrepancy'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title(sprintf('Regularization Parameter for Simulated Data - MTSVD'),'FontSize
    ',12)
axis([avec(1) avec(end) 0 1]);
figure(figure9)
%-------------------------------------------------------------------------

[RD mnrdix] = min(OP(2:end,1));
lam = OP(mnrdix+1,2)
%    pause
end
```

A.3 MATLAB CODE 3 - Forecasting with early data - no confidence intervals

```
function beta_line_forecasting_OPT_TD_SL
```

```
% This function produces forecasts for remaining outbreak time periods
% using early data in steps. All three regularization methods are
% employed. This function produces a single plot forecast based on the
% recovered transmission rate. This function is set up for data from
% Sierra Leone, however other data sets may be analyzed. Note that
% regularization parameter ranges must be determined for each data set
% before it can be effectively applied. The subroutine for regularization
% parameter selection provides plots of the behavior of the regularization
% parameter for use.

close all
clear all
clc
format long
warning('off','all')

global N kappa gamma j C0 E0 I0 S0 mub
%-------------------------------------------------------------------------
% Pull Data and Set Program Variables
load SLcumcases.txt;
tdata_full = SLcumcases(:,1);
Cdata_full = SLcumcases(:,2);
Skip = 5;
Start = 6;
N = 6e6; % Sierra Leone;
lambdaF = 2e-5; %SL
CVS = 5;
WW = Start:Skip:(Skip*(CVS-1)+Start)
RegID = 'Sierra Leone';
kappa = 7/8;
gamma = 7/6;
Cdata_inc_full = [Cdata_full(1,1); diff(Cdata_full(:,1))];
S0 = N;
C0 = Cdata_full(1,1);
```

```
E0 = Cdata_inc_full(1,1)/kappa;
I0 = C0;
M = length(tdata_full);
mub = M;
count = 0;
%-------------------------------------------------------------------------
% Iterate through early data sets
for j = Start:Skip:WW(CVS) % SL
    count = count + 1
    j
    tdata = tdata_full(1:j,1);
    Cdata = Cdata_full(1:j,1);
    Cdata_inc = Cdata_inc_full(1:j,1);
    n = ceil(2*j);
    %-------------------------------------------------------------------------
    % Construct A
    A = zeros(j-1,n+1);
    [f,K] = kernel(Cdata_inc, Cdata, tdata);
    K_sp = spline(tdata, K);
    a = tdata(1); b = tdata_full(mub);
    for k1 = 1:n+1
        for k2 = 1:j-1
            A(k2,k1) = integral(@(x) leg(x, k1-1, a, b).*ppval(K_sp,x),...
                tdata(1), tdata(k2+1));
        end
    end
    % end Construct A
    condition = cond(A)
    %-------------------------------------------------------------------------
    % Subroutines for Regularization Parameter Selection All Methods
    for Meth = 1:3
        [lambda RD] = RelDisc(A,f,lambdaF,Cdata_inc,n,tdata_full,Meth,count);
        RDM(Meth+(count-1)*3,1:2) = [lambda, RD]
```

```
switch Meth
    case 1
        % MTSVD
        [U,S,V] = svd(A);
        s = diag(S);
        PIS = zeros(j-1,n+1);
        for i = 1:j-1
            if s(i) >= lambda
                PIS(i,i) = 1/s(i);
            else
                PIS(i,i) = 1/lambda;
            end
        end
        PIA = V*(PIS)'*U';
        Coef = PIA*f(2:j,1)
    case 2
        %TSVD
        [U,S,V] = svd(A'*A);
        s = diag(S);
        sinv = 1./s;
        for i = 1:n+1
            if s(i)< lambda
                sinv(i) = 0;
            end
        end
        PIS = diag(sinv);
        PIA = V*PIS*U';
        Coef = PIA*A'*f(2:j,1);
    case 3
        % TIK
         Coef = (A'*A + lambda*eye(n+1))\(A'*f(2:j,1));
end

%-------------------------------------------------------------------------
```

```
% Assemble Vectors for Plotting
switch Meth
    case 1
        if count == 1
        [x1M, y1M] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
            tdata(1) tdata(j)],'-k');
        [x1aM, y1aM] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
            mub)),[tdata_full(j) tdata_full(mub)],'--k');

        [T, F] = operator(tdata_full,Coef,n);
        F_sp = spline(T, F);
        [T1M, F1M] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-k');
        [T1aM, F1aM] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
            mub)],'--k')
        FCSP = ppval(spline(T1aM,F1aM),(tdata_full(j)+4):(tdata_full(j)+6)
            )
        %pause

        end
        if count == 2
        [x2M, y2M] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
            tdata(1) tdata(j)],'-r');
        [x2aM, y2aM] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
            mub)),[tdata_full(j) tdata_full(mub)],'--r');

        [T, F] = operator(tdata_full,Coef,n);
        F_sp = spline(T, F);
        [T2M, F2M] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-r');
        [T2aM, F2aM] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
            mub)],'--r')
        FCSP = ppval(spline(T2aM,F2aM),(tdata_full(j)+4):(tdata_full(j)+6)
            )
        %pause
```

```
end
if count == 3
[x3M, y3M] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
    tdata(1) tdata(j)],'-c');
[x3aM, y3aM] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
    mub)),[tdata_full(j) tdata_full(mub)],'--c');

[T, F] = operator(tdata_full,Coef,n);
F_sp = spline(T, F);
[T3M, F3M] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-c');
[T3aM, F3aM] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
    mub)],'--c')
FCSP = ppval(spline(T3aM,F3aM),(tdata_full(j)+4):(tdata_full(j)+6)
    )
%pause

end
if count == 4
[x4M, y4M] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
    tdata(1) tdata(j)],'-b');
[x4aM, y4aM] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
    mub)),[tdata_full(j) tdata_full(mub)],'--b');

[T, F] = operator(tdata_full,Coef,n);
F_sp = spline(T, F);
[T4M, F4M] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-b');
[T4aM, F4aM] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
    mub)],'--b')
FCSP = ppval(spline(T4aM,F4aM),(tdata_full(j)+4):(tdata_full(j)+6)
    )
%pause

end
if count == 5
```

```
        [x5M, y5M] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
            tdata(1) tdata(j)],'-g');
        [x5aM, y5aM] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
            mub)),[tdata_full(j) tdata_full(mub)],'--g');

        [T, F] = operator(tdata_full,Coef,n);
        F_sp = spline(T, F);
        [T5M, F5M] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-g');
        [T5aM, F5aM] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
            mub)],'--g')
        FCSP = ppval(spline(T5aM,F5aM),(tdata_full(j)+4):(tdata_full(j)+6)
            )
        %pause

        end

    case 2
        if count == 1
        [x1T, y1T] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
            tdata(1) tdata(j)],'-k');
        [x1aT, y1aT] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
            mub)),[tdata_full(j) tdata_full(mub)],'--k');

        [T, F] = operator(tdata_full,Coef,n);
        F_sp = spline(T, F);
        [T1T, F1T] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-k');
        [T1aT, F1aT] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
            mub)],'--k');

        end
        if count == 2
        [x2T, y2T] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
            tdata(1) tdata(j)],'-r');
```

```
[x2aT, y2aT] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
    mub)),[tdata_full(j) tdata_full(mub)],'--r');

[T, F] = operator(tdata_full,Coef,n);
F_sp = spline(T, F);
[T2T, F2T] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-r');
[T2aT, F2aT] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
    mub)],'--r');

end
if count == 3
[x3T, y3T] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
    tdata(1) tdata(j)],'-c');
[x3aT, y3aT] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
    mub)),[tdata_full(j) tdata_full(mub)],'--c');

[T, F] = operator(tdata_full,Coef,n);
F_sp = spline(T, F);
[T3T, F3T] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-c');
[T3aT, F3aT] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
    mub)],'--c');

end
if count == 4
[x4T, y4T] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
    tdata(1) tdata(j)],'-b');
[x4aT, y4aT] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
    mub)),[tdata_full(j) tdata_full(mub)],'--b');

[T, F] = operator(tdata_full,Coef,n);
F_sp = spline(T, F);
[T4T, F4T] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-b');
[T4aT, F4aT] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
    mub)],'--b');
```

```
        end
        if count == 5
        [x5T, y5T] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
            tdata(1) tdata(j)],'-g');
        [x5aT, y5aT] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
            mub)),[tdata_full(j) tdata_full(mub)],'--g');

        [T, F] = operator(tdata_full,Coef,n);
        F_sp = spline(T, F);
        [T5T, F5T] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-g');
        [T5aT, F5aT] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
            mub)],'--g');

        end

    case 3
        if count == 1
        [x1K, y1K] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
            tdata(1) tdata(j)],'-k');
        [x1aK, y1aK] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
            mub)),[tdata_full(j) tdata_full(mub)],'--k');

        [T, F] = operator(tdata_full,Coef,n);
        F_sp = spline(T, F);
        [T1K, F1K] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-k');
        [T1aK, F1aK] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
            mub)],'--k');

        end
        if count == 2
        [x2K, y2K] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
            tdata(1) tdata(j)],'-r');
```

```
[x2aK, y2aK] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
    mub)),[tdata_full(j) tdata_full(mub)],'--r');

[T, F] = operator(tdata_full,Coef,n);
F_sp = spline(T, F);
[T2K, F2K] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-r');
[T2aK, F2aK] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
    mub)],'--r');

end
if count == 3
[x3K, y3K] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
    tdata(1) tdata(j)],'-c');
[x3aK, y3aK] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
    mub)),[tdata_full(j) tdata_full(mub)],'--c');

[T, F] = operator(tdata_full,Coef,n);
F_sp = spline(T, F);
[T3K, F3K] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-c');
[T3aK, F3aK] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
    mub)],'--c');

end
if count == 4
[x4K, y4K] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
    tdata(1) tdata(j)],'-b');
[x4aK, y4aK] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
    mub)),[tdata_full(j) tdata_full(mub)],'--b');

[T, F] = operator(tdata_full,Coef,n);
F_sp = spline(T, F);
[T4K, F4K] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-b');
[T4aK, F4aK] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
    mub)],'--b');
```

```
            end
            if count == 5
            [x5K, y5K] = fplot(@(x)approx(x,n,Coef,tdata(1),tdata_full(mub)),[
                tdata(1) tdata(j)],'-g');
            [x5aK, y5aK] = fplot(@(x)approx(x,n,Coef,tdata_full(1),tdata_full(
                mub)),[tdata_full(j) tdata_full(mub)],'--g');

            [T, F] = operator(tdata_full,Coef,n);
            F_sp = spline(T, F);
            [T5K, F5K] = fplot(@(x) ppval(F_sp,x),[tdata(1) tdata(j)],'-g');
            [T5aK, F5aK] = fplot(@(x) ppval(F_sp,x),[tdata_full(j) tdata_full(
                mub)],'--g');

            end

    end
    %-------------------------------------------------------------------------

    end  % end Method cycle
end  % end early data cycle

%-------------------------------------------------------------------------
% Plots by Method
for Meth = 1:3
    switch Meth
        case 1
        figure2 = figure;
        axes2 = axes('Parent',figure2,...
            'AmbientLightColor',[0.941176470588235 0.941176470588235
                0.941176470588235]);
        box(axes2,'on');
        hold(axes2,'all');
        switch CVS
```

```
        case 5
            plot(T1M, F1M, '-k',T2M, F2M, '-c',T3M, F3M, '-b',T4M, F4M, '-
                g',T5M, F5M, '-m', tdata_full, Cdata_inc_full, '*r',...
                T1aM, F1aM, '--k',T2aM, F2aM, '--c',T3aM, F3aM, '--b',T4aM
                    , F4aM, '--g',T5aM, F5aM, '--m', 'linewidth',2);
            legend({sprintf('Data, %d weeks',WW(1)),sprintf('Data, %d
                weeks',WW(2)),sprintf('Data, %d weeks',WW(3)),...
                sprintf('Data, %d weeks',WW(4)),sprintf('Data, %d weeks',
                    WW(5)),'Real Data'},...
                'FontSize',12,'Location','best')
        case 4
            plot(T1M, F1M, '-k',T2M, F2M, '-c',T3M, F3M, '-b',T4M, F4M, '-
                g', tdata_full, Cdata_inc_full, '*r',...
                T1aM, F1aM, '--k',T2aM, F2aM, '--c',T3aM, F3aM, '--b',T4aM
                    , F4aM, '--g', 'linewidth',2);
            legend({sprintf('Data, %d days',WW(1)),sprintf('Data, %d days'
                ,WW(2)),sprintf('Data, %d days',WW(3)),...
                sprintf('Data, %d days',WW(4)),'Real Data'},...
                'FontSize',12,'Location','best')
    end
    axis([0 M 0 max(Cdata_inc_full)*1.5])
    xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
        Computer Modern');
    ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
        Computer Modern');
    title(sprintf('Incidence Data Generated by Recovered \\beta - - %s -
        MTSVD', RegID),'FontSize',12)
    figure(figure2)
case 2
    figure4 = figure;
    axes2 = axes('Parent',figure4,...
        'AmbientLightColor',[0.941176470588235 0.941176470588235
            0.941176470588235]);
    box(axes2,'on');
```

```
    hold(axes2,'all');
    switch CVS
        case 5
            plot(T1T, F1T, '-k',T2T, F2T, '-c',T3T, F3T, '-b',T4T, F4T, '-
                g',T5T, F5T, '-m', tdata_full, Cdata_inc_full, '*r',...
                 T1aT, F1aT, '--k',T2aT, F2aT, '--c',T3aT, F3aT, '--b',T4aT
                     , F4aT, '--g',T5aT, F5aT, '--m', 'linewidth',2);
            legend({sprintf('Data, %d weeks',WV(1)),sprintf('Data, %d
                weeks',WV(2)),sprintf('Data, %d weeks',WV(3)),...
                 sprintf('Data, %d weeks',WV(4)),sprintf('Data, %d weeks',
                     WV(5)),'Real Data'},...
                 'FontSize',12,'Location','best')
        case 4
            plot(T1T, F1T, '-k',T2T, F2T, '-c',T3T, F3T, '-b',T4T, F4T, '-
                g', tdata_full, Cdata_inc_full, '*r',...
                 T1aT, F1aT, '--k',T2aT, F2aT, '--c',T3aT, F3aT, '--b',T4aT
                     , F4aT, '--g', 'linewidth',2);
            legend({sprintf('Data, %d days',WV(1)),sprintf('Data, %d days'
                ,WV(2)),sprintf('Data, %d days',WV(3)),...
                 sprintf('Data, %d days',WV(4)),'Real Data'},...
                 'FontSize',12,'Location','best')
    end
    axis([0 M 0 max(Cdata_inc_full)*1.5])
    xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
        Computer Modern');
    ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
        Computer Modern');
    title(sprintf('Incidence Data Generated by Recovered \\beta - - %s -
        TSVD', RegID),'FontSize',12)
    figure(figure4)
case 3
    figure6 = figure;
    axes2 = axes('Parent',figure6,...
```

```
        'AmbientLightColor',[0.941176470588235 0.941176470588235
            0.941176470588235]);
    box(axes2,'on');
    hold(axes2,'all');
    switch CVS
        case 5
            plot(T1K, F1K, '-k',T2K, F2K, '-c',T3K, F3K, '-b',T4K, F4K, '-
                g',T5K, F5K, '-m', tdata_full, Cdata_inc_full, '*r',...
                 T1aK, F1aK, '--k',T2aK, F2aK, '--c',T3aK, F3aK, '--b',T4aK
                     , F4aK, '--g',T5aK, F5aK, '--m', 'linewidth',2);
            legend({sprintf('Data, %d weeks',WV(1)),sprintf('Data, %d
                weeks',WV(2)),sprintf('Data, %d weeks',WV(3)),...
                 sprintf('Data, %d weeks',WV(4)),sprintf('Data, %d weeks',
                    WV(5)),'Real Data'},...
                 'FontSize',12,'Location','best')
        case 4
            plot(T1K, F1K, '-k',T2K, F2K, '-c',T3K, F3K, '-b',T4K, F4K, '-
                g', tdata_full, Cdata_inc_full, '*r',...
                 T1aK, F1aK, '--k',T2aK, F2aK, '--c',T3aK, F3aK, '--b',T4aK
                     , F4aK, '--g', 'linewidth',2);
            legend({sprintf('Data, %d days',WV(1)),sprintf('Data, %d days'
                ,WV(2)),sprintf('Data, %d days',WV(3)),...
                 sprintf('Data, %d days',WV(4)),'Real Data'},...
                 'FontSize',12,'Location','best')
    end
    axis([0 M 0 max(Cdata_inc_full)*1.5])
    xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
        Computer Modern');
    ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
        Computer Modern');
    title(sprintf('Incidence Data Generated by Recovered \\beta - - %s -
        Tikhonov', RegID),'FontSize',12)
    figure(figure6)
end
```

```
end
%-------------------------------------------------------------------------
% Plots Assembled
        figure7 = figure;
        axes2 = axes('Parent',figure7,...
            'AmbientLightColor',[0.941176470588235 0.941176470588235
                0.941176470588235]);
        box(axes2,'on');
        hold(axes2,'all');
        plot(T3M, F3M, '-b', T3T, F3T, '-g', T3K, F3K, '-m', tdata_full,
            Cdata_inc_full, '*r',...
            T3aM, F3aM, '--b', T3aT, F3aT, '--g', T3aK, F3aK, '--m', '
                linewidth', 2);
        legend({'MTSVD', 'TSVD', 'Tikhonov', 'Real Data'}, 'FontSize',12, '
            Location', 'best')
        axis([0 M 0 max(Cdata_inc_full)*1.5])
        xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
            Computer Modern');
        ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
            Computer Modern');
        title({sprintf('Incidence Data and Projection Generated by Recovered
            \\beta - %s - All Methods', RegID);sprintf(' %2.0f Weeks',WW(3))
            },'FontSize',12)
        figure(figure7)
        figure8 = figure;
        axes2 = axes('Parent',figure8,...
            'AmbientLightColor',[0.941176470588235 0.941176470588235
                0.941176470588235]);
        box(axes2,'on');
        hold(axes2,'all');
        plot(T4M, F4M, '-b', T4T, F4T, '-g', T4K, F4K, '-m', tdata_full,
            Cdata_inc_full, '*r',...
```

```
    T4aM, F4aM, '--b', T4aT, F4aT, '--g', T4aK, F4aK, '--m', '
        linewidth', 2);
legend({'MTSVD', 'TSVD', 'Tikhonov', 'Real Data'}, 'FontSize',12, '
    Location', 'best')
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title({sprintf('Incidence Data and Projection Generated by Recovered
    \\beta - %s - All Methods', RegID);sprintf('  %2.0f Weeks',WV(4))
    },'FontSize',12)
figure(figure8)

%MTSVD TD
figure9 = figure;
axes2 = axes('Parent',figure9,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1M, F1M, '-k', tdata_full, Cdata_inc_full, '*r',...
            T1aM, F1aM, '--k', 'linewidth',2);
        legend({sprintf('Data, %d weeks',WV(1)),'Real Data'},...
            'FontSize',12,'Location','best')
    case 4
        plot(T1M, F1M, '-k', tdata_full,Cdata_inc_full, '*r',...
            T1aM, F1aM, '--k', 'linewidth',2);
        legend({sprintf('Data, %d days',WV(1)),'Real Data'},...
            'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
```

```
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title(sprintf('Incidence Data Generated by Recovered \\beta - - %s - -
    MTSVD', RegID),'FontSize',12)
figure(figure9)

figure10 = figure;
axes2 = axes('Parent',figure10,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1M, F1M, '-k',T2M, F2M, '-c', tdata_full, Cdata_inc_full
            , '*r',...
             T1aM, F1aM, '--k',T2aM, F2aM, '--c', 'linewidth',2);
        legend({sprintf('Data, %d weeks',WV(1)),sprintf('Data, %d
            weeks',WV(2)),'Real Data'},...
             'FontSize',12,'Location','best')
    case 4
        plot(T1M, F1M, '-k',T2M, F2M, '-c', tdata_full, Cdata_inc_full
            , '*r',...
             T1aM, F1aM, '--k',T2aM, F2aM, '--c', 'linewidth',2);
        legend({sprintf('Data, %d days',WV(1)),sprintf('Data, %d days'
            ,WV(2)),'Real Data'},...
             'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
```

```
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title(sprintf('Incidence Data Generated by Recovered \\beta - - %s - '
    MTSVD', RegID),'FontSize',12)
figure(figure10)

figure11 = figure;
axes2 = axes('Parent',figure11,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1M, F1M, '-k',T2M, F2M, '-c',T3M, F3M, '-b', tdata_full,
            Cdata_inc_full, '*r',...
            T1aM, F1aM, '--k',T2aM, F2aM, '--c',T3aM, F3aM, '--b', '
                linewidth',2);
        legend({sprintf('Data, %d weeks',WV(1)),sprintf('Data, %d
            weeks',WV(2)),sprintf('Data, %d weeks',WV(3)),'Real Data'
            },...
            'FontSize',12,'Location','best')
    case 4
        plot(T1M, F1M, '-k',T2M, F2M, '-c',T3M, F3M, '-b', tdata_full,
            Cdata_inc_full, '*r',...
            T1aM, F1aM, '--k',T2aM, F2aM, '--c',T3aM, F3aM, '--b', '
                linewidth',2);
        legend({sprintf('Data, %d days',WV(1)),sprintf('Data, %d days'
            ,WV(2)),sprintf('Data, %d days',WV(3)),'Real Data'},...
            'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
```

```
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title(sprintf('Incidence Data Generated by Recovered \\beta = %s - 
    MTSVD', RegID),'FontSize',12)
figure(figure11)

figure12 = figure;
axes2 = axes('Parent',figure12,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1M, F1M, '-k',T2M, F2M, '-c',T3M, F3M, '-b',T4M, F4M, '-
            g', tdata_full, Cdata_inc_full, '*r',...
             T1aM, F1aM, '--k',T2aM, F2aM, '--c',T3aM, F3aM, '--b',T4aM
                 , F4aM, '--g', 'linewidth',2);
        legend({sprintf('Data, %d weeks',WW(1)),sprintf('Data, %d 
            weeks',WW(2)),sprintf('Data, %d weeks',WW(3)),...
             sprintf('Data, %d weeks',WW(4)),'Real Data'},...
             'FontSize',12,'Location','best')
    case 4
        plot(T1M, F1M, '-k',T2M, F2M, '-c',T3M, F3M, '-b',T4M, F4M, '-
            g', tdata_full, Cdata_inc_full, '*r',...
             T1aM, F1aM, '--k',T2aM, F2aM, '--c',T3aM, F3aM, '--b',T4aM
                 , F4aM, '--g', 'linewidth',2);
        legend({sprintf('Data, %d days',WW(1)),sprintf('Data, %d days'
            ,WW(2)),sprintf('Data, %d days',WW(3)),...
             sprintf('Data, %d days',WW(4)),'Real Data'},...
             'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
```

```
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title(sprintf('Incidence Data Generated by Recovered \\beta - - %s - 
    MTSVD', RegID),'FontSize',12)
figure(figure12)

figure13 = figure;
axes2 = axes('Parent',figure13,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1M, F1M, '-k',T2M, F2M, '-c',T3M, F3M, '-b',T4M, F4M, '-
            g',T5M, F5M, '-m', tdata_full, Cdata_inc_full, '*r',...
             T1aM, F1aM, '--k',T2aM, F2aM, '--c',T3aM, F3aM, '--b',T4aM
                 , F4aM, '--g',T5aM, F5aM, '--m', 'linewidth',2);
        legend({sprintf('Data, %d weeks',WW(1)),sprintf('Data, %d
            weeks',WW(2)),sprintf('Data, %d weeks',WW(3)),...
             sprintf('Data, %d weeks',WW(4)),sprintf('Data, %d weeks',
                 WW(5)),'Real Data'},...
             'FontSize',12,'Location','best')
    case 4
        plot(T1M, F1M, '-k',T2M, F2M, '-c',T3M, F3M, '-b',T4M, F4M, '-
            g', tdata_full, Cdata_inc_full, '*r',...
             T1aM, F1aM, '--k',T2aM, F2aM, '--c',T3aM, F3aM, '--b',T4aM
                 , F4aM, '--g', 'linewidth',2);
        legend({sprintf('Data, %d days',WW(1)),sprintf('Data, %d days'
            ,WW(2)),sprintf('Data, %d days',WW(3)),...
             sprintf('Data, %d days',WW(4)),'Real Data'},...
             'FontSize',12,'Location','best')
```

```
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title(sprintf('Incidence Data Generated by Recovered \\beta - - %s - '
    MTSVD', RegID),'FontSize',12)
figure(figure13)

%Tikhonov TD
figure14 = figure;
axes2 = axes('Parent',figure14,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1K, F1K, '-k', tdata_full, Cdata_inc_full, '*r',...
            T1aK, F1aK, '--k', 'linewidth',2);
        legend({sprintf('Data, %d weeks',WV(1)),'Real Data'},...
            'FontSize',12,'Location','best')
    case 4
        plot(T1K, F1K, '-k', tdata_full,Cdata_inc_full, '*r',...
            T1aK, F1aK, '--k', 'linewidth',2);
        legend({sprintf('Data, %d days',WV(1)),'Real Data'},...
            'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
```

```
title(sprintf('Incidence Data Generated by Recovered \\beta - - %s - ...
    Tikhonov', RegID),'FontSize',12)
figure(figure14)

figure15 = figure;
axes2 = axes('Parent',figure15,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235 ...
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1K, F1K, '-k',T2K, F2K, '-c', tdata_full, Cdata_inc_full ...
            , '*r',...
            T1aK, F1aK, '--k',T2aK, F2aK, '--c', 'linewidth',2);
        legend({sprintf('Data, %d weeks',WV(1)),sprintf('Data, %d ...
            weeks',WV(2)),'Real Data'},...
            'FontSize',12,'Location','best')
    case 4
        plot(T1K, F1K, '-k',T2K, F2K, '-c', tdata_full, Cdata_inc_full ...
            , '*r',...
            T1aK, F1aK, '--k',T2aK, F2aK, '--c', 'linewidth',2);
        legend({sprintf('Data, %d days',WV(1)),sprintf('Data, %d days' ...
            ,WV(2)),'Real Data'},...
            'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','...
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','...
    Computer Modern');
title(sprintf('Incidence Data Generated by Recovered \\beta - - %s - ...
    Tikhonov', RegID),'FontSize',12)
figure(figure15)
```

```
figure16 = figure;
axes2 = axes('Parent',figure16,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1K, F1K, '-k',T2K, F2K, '-c',T3K, F3K, '-b', tdata_full,
            Cdata_inc_full, '*r',...
            T1aK, F1aK, '--k',T2aK, F2aK, '--c',T3aK, F3aK, '--b', '
                linewidth',2);
        legend({sprintf('Data, %d weeks',WV(1)),sprintf('Data, %d
            weeks',WV(2)),sprintf('Data, %d weeks',WV(3)),'Real Data'
            },...
            'FontSize',12,'Location','best')
    case 4
        plot(T1K, F1K, '-k',T2K, F2K, '-c',T3K, F3K, '-b', tdata_full,
            Cdata_inc_full, '*r',...
            T1aK, F1aK, '--k',T2aK, F2aK, '--c',T3aK, F3aK, '--b', '
                linewidth',2);
        legend({sprintf('Data, %d days',WV(1)),sprintf('Data, %d days'
            ,WV(2)),sprintf('Data, %d days',WV(3)),'Real Data'},...
            'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title(sprintf('Incidence Data Generated by Recovered \\beta - - %s -
    Tikhonov', RegID),'FontSize',12)
figure(figure16)
```

```
figure17 = figure;
axes2 = axes('Parent',figure17,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1K, F1K, '-k',T2K, F2K, '-c',T3K, F3K, '-b',T4K, F4K, '-
            g', tdata_full, Cdata_inc_full, '*r',...
             T1aK, F1aK, '--k',T2aK, F2aK, '--c',T3aK, F3aK, '--b',T4aK
                 , F4aK, '--g', 'linewidth',2);
        legend({sprintf('Data, %d weeks',WW(1)),sprintf('Data, %d
            weeks',WW(2)),sprintf('Data, %d weeks',WW(3)),...
             sprintf('Data, %d weeks',WW(4)),'Real Data'},...
             'FontSize',12,'Location','best')
    case 4
        plot(T1K, F1K, '-k',T2K, F2K, '-c',T3K, F3K, '-b',T4K, F4K, '-
            g', tdata_full, Cdata_inc_full, '*r',...
             T1aK, F1aK, '--k',T2aK, F2aK, '--c',T3aK, F3aK, '--b',T4aK
                 , F4aK, '--g', 'linewidth',2);
        legend({sprintf('Data, %d days',WW(1)),sprintf('Data, %d days'
            ,WW(2)),sprintf('Data, %d days',WW(3)),...
             sprintf('Data, %d days',WW(4)),'Real Data'},...
             'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
title(sprintf('Incidence Data Generated by Recovered \\beta - %s -
    Tikhonov', RegID),'FontSize',12)
```

```
figure(figure12)

figure13 = figure;
axes2 = axes('Parent',figure13,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
switch CVS
    case 5
        plot(T1K, F1K, '-k',T2K, F2K, '-c',T3K, F3K, '-b',T4K, F4K, '-
            g',T5K, F5K, '-m', tdata_full, Cdata_inc_full, '*r',...
             T1aK, F1aK, '--k',T2aK, F2aK, '--c',T3aK, F3aK, '--b',T4aK
                 , F4aK, '--g',T5aK, F5aK, '--m', 'linewidth',2);
        legend({sprintf('Data, %d weeks',WV(1)),sprintf('Data, %d
            weeks',WV(2)),sprintf('Data, %d weeks',WV(3)),...
             sprintf('Data, %d weeks',WV(4)),sprintf('Data, %d weeks',
                 WV(5)),'Real Data'},...
             'FontSize',12,'Location','best')
    case 4
        plot(T1K, F1K, '-k',T2K, F2K, '-c',T3K, F3K, '-b',T4K, F4K, '-g
            ', tdata_full, Cdata_inc_full, '*r',...
             T1aK, F1aK, '--k',T2aK, F2aK, '--c',T3aK, F3aK, '--b',T4aK
                 , F4aK, '--g', 'linewidth',2);
        legend({sprintf('Data, %d days',WV(1)),sprintf('Data, %d days'
            ,WV(2)),sprintf('Data, %d days',WV(3)),...
             sprintf('Data, %d days',WV(4)),'Real Data'},...
             'FontSize',12,'Location','best')
end
axis([0 M 0 max(Cdata_inc_full)*1.5])
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
ylabel({'Incidence Data'},'LineWidth',2,'FontSize',12,'FontName','
    Computer Modern');
```

```
        title(sprintf('Incidence Data Generated by Recovered \\beta - - %s - '
            Tikhonov', RegID),'FontSize',12)
        figure(figure13)
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [f,K] = kernel(Cdata_inc, Cdata, tdata)
global N kappa gamma j C0 E0 I0 S0
K = zeros(j,1);
f = zeros(j,1);

S_p = spline(tdata(1:j,1),Cdata_inc(1:j,1));
It = zeros(j,1);

for i = 1:j
      It(i,1) = integral(@(t) exp(-gamma*(tdata(i,1) - t)).*ppval(S_p,t),
          tdata(1), tdata(i));
      K(i,1) = I0*exp(-gamma*(tdata(i,1) - tdata(1,1))) + It(i,1);
      f(i,1) = - log((- Cdata_inc(i,1)/kappa - Cdata(i,1)+ E0 + C0)/S0 + 1);
end

K = K/N;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    function P = leg(x, k, a, b)
        t = (2.*x - a - b)./(b - a);
                if k == 0
                    P1 = 1; P = P1;
```

```
        elseif k == 1
            P2 = t; P = P2;
        else
            P1 = 1; P2 = t;
            for i = 2:k
          P3 = ((2*(i-1)+1).*t.*P2 - (i-1).*P1)./i;
          P1 = P2; P2 = P3;
            end
            P = P3;
        end

    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = approx(x,m,c,a,b)
    z = 0;
       for k = 1:m+1
       z = z + c(k).*leg(x, k-1, a, b);
       end
       if z < 0
           z = 0;
       end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function dy = sstm(x,y,Coef,tdata_full,number)
global N kappa gamma mub

dy = zeros(3,1);
dy(1) = -approx(x,number,Coef,tdata_full(1),tdata_full(mub)).*y(1).*y(3)/N;
dy(2) = approx(x,number,Coef,tdata_full(1),tdata_full(mub)).*y(1).*y(3)/N -
    kappa*y(2);
dy(3) = kappa*y(2) - gamma*y(3);
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [T,F] = operator(tdata_full,Coef,number)
global S0 E0 I0 kappa mub

options = odeset('RelTol',1e-4,'AbsTol',1e-6);
[T,Y] = ode23s(@(x,y) sstm(x,y,Coef,tdata_full,number),tdata_full(1:mub,1),[S0
    E0 I0], options);

F = kappa*Y(:,2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [lam RD] = RelDisc(A,f,temp,Cdata_inc,n,tdata_full,Meth,count)
global j
Lmcount = 1;
OP = [];
switch Meth
    case 1
        %MTSVD
        [U,S,V] = svd(A);
        s = diag(S);
        avec = linspace(1e-8,5.5e-5,50);
        for pw = 1:length(avec)
            lambda = avec(pw);
            PIS = zeros(j-1,n+1);
            for i = 1:j-1
                if s(i) >= lambda
                    PIS(i,i) = 1/s(i);
                else
                    PIS(i,i) = 1/lambda;
```

```
            end
        end
        PIA = V*(PIS)'*U';
        Coef = PIA*f(2:j,1);
        [Tt, Ft] = operator(tdata_full,Coef,n);
        RD = norm(Ft(1:j)-Cdata_inc)/norm(Cdata_inc);
        OP(Lmcount,1:2) = [RD lambda];
        Lmcount = Lmcount + 1;
    end
    OP
    figure9 = figure;
    axes2 = axes('Parent',figure9,...
        'AmbientLightColor',[0.941176470588235 0.941176470588235
            0.941176470588235]);
    box(axes2,'on');
    hold(axes2,'all');
    LAM = OP(:,2);
    RDis = OP(:,1);
    plot(LAM,RDis,'b','LineWidth',2)
    legend({sprintf('MTSVD')})%,'Interpreter','latex')
    xlabel({'\alpha'},'LineWidth',2,'FontSize',12,'FontName','Computer␣
        Modern');
    ylabel({'Relative␣Discrepancy'},'LineWidth',2,'FontSize',12,'FontName'
        ,'Computer␣Modern');
    title(sprintf('Regularization␣Parameter␣for␣Simulated␣Data␣-␣MTSVD'),'
        FontSize',12)
    axis([avec(1) avec(end) 0 1]);
    figure(figure9)
    [RD mnrdix] = min(OP(2:end,1));
    lam = OP(mnrdix+1,2)

case 2
    %TSVD
    [U,S,V] = svd(A'*A);
```

```
s = diag(S)
avec = linspace(1e-15,.3e-8,50);
for pw = 1:length(avec)
    lambda = avec(pw);
    sinv = 1./s;
    for i = 1:n+1
        if s(i)<= lambda
            sinv(i) = 0;
        end
    end
    PIS = diag(sinv);
    PIA = V*PIS*U';
    Coef = PIA*A'*f(2:j,1);
    [Tt, Ft] = operator(tdata_full,Coef,n);
    RD = norm(Ft(1:j)-Cdata_inc)/norm(Cdata_inc);
    OP(Lmcount,1:2) = [RD lambda];
    Lmcount = Lmcount + 1;
end
OP
figure10 = figure;
axes2 = axes('Parent',figure10,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
LAM = OP(:,2);
RDis = OP(:,1);
plot(LAM,RDis,'g','LineWidth',2)
legend({sprintf('TSVD')})%,'Interpreter','latex')
xlabel({'\alpha'},'LineWidth',2,'FontSize',12,'FontName','Computer␣
    Modern');
ylabel({'Relative␣Discrepancy'},'LineWidth',2,'FontSize',12,'FontName'
    ,'Computer␣Modern');
```

```
    title(sprintf('Regularization Parameter for Simulated Data - TSVD'),'
        FontSize',12)
    axis([avec(1) avec(end) 0 1]);
    figure(figure10)
    [RD mnrdix] = min(OP(2:end,1));
    lam = OP(mnrdix+1,2)

case 3
    %Tik
    avec = linspace(1e-15,.3e-8,50);
    for pw = 1:length(avec)
        lambda = avec(pw);
        Coef = (A'*A + lambda*eye(n+1))\(A'*f(2:j,1));
        [Tt, Ft] = operator(tdata_full,Coef,n);
        RD = norm(Ft(1:j)-Cdata_inc)/norm(Cdata_inc);
        OP(Lmcount,1:2) = [RD lambda];
        Lmcount = Lmcount + 1;
    end
    OP
    figure11 = figure;
    axes2 = axes('Parent',figure11,...
        'AmbientLightColor',[0.941176470588235 0.941176470588235
            0.941176470588235]);
    box(axes2,'on');
    hold(axes2,'all');
    LAM = OP(:,2);
    RDis = OP(:,1);
    plot(LAM,RDis,'m','LineWidth',2)
    legend({sprintf('Tikhonov')})%,'Interpreter','latex')
    xlabel({'\alpha'},'LineWidth',2,'FontSize',12,'FontName','Computer
        Modern');
    ylabel({'Relative Discrepancy'},'LineWidth',2,'FontSize',12,'FontName'
        ,'Computer Modern');
```

```
        title(sprintf('Regularization Parameter for Simulated Data - Tikhonov'
            ),'FontSize',12)
        axis([avec(1) avec(end) 0 1]);
        figure(figure11)
        [RD mnrdix] = min(OP(2:end,1));
        lam = OP(mnrdix+1,2)

end

end
```

A.4 MATLAB CODE 4 - Forecasting with early data

```
function beta_linear_MTSVD_FC_DP_Uncertainty_GenFiles2
%from beta_line_MTSVD_forcasting_SL_OPT_DP2_UncertaintyC
% This function generates data files for plotting confidence intervals
% using early data.  The 8 data sets used are available.  Poisson curves
% are generated for short term data sets. From Poisson curve, beta is
% recovered and used to generate the remaining weeks of data.  These data
% files are used to plot confidence intervals.

close all
clear all
clc
format long
warning('off','all')

global N kappa gamma j C0 E0 I0 S0 mub
%-------------------------------------------------------------------------
% USER INPUT
DSet = str2double(input(['Choose from among the following data sets by number
    indicated',...
```

```
        '\n 1) Sierra Leone',...
        '\n 2) Liberia',...
        '\n 3) Western Area Rural',...
        '\n 4) Western Area Urban',...
        '\n 5) Montserrado',...
        '\n 6) Gueckedou',...
        '\n 7) London Measles',...
        '\n 8) San Fransisco Pandemic Influenza',...
        '\n',...
        '\n Which?  '], 's'));
while isnan(DSet) || fix(DSet) ~= DSet || DSet<1 ||DSet>8
    DSet = str2double(input('Please enter and INTEGER between 1 and 8: ', 's')
        );
end
   switch DSet
       case 1
           load SLcumcases.txt;
           tdata_full = SLcumcases(:,1);
           Cdata_full = SLcumcases(:,2);
           Skip = 5;
           Start = 11;
           N = 6e6; % Sierra Leone;
           lambdaF = 2e-7; %SL
           CVS = 4;
           WV = Start:Skip:(Skip*(CVS-1)+Start)
           RegID = 'Sierra Leone';
           kappa = 7/8;
           gamma = 7/6;
       case 2
           load LIBcumcases_rev.txt;
           tdata_full = LIBcumcases_rev(:,1);
           Cdata_full = LIBcumcases_rev(:,2);
           Skip = 2;
           Start = 13;
```

```
    N = 4e6; % Liberia
    lambdaF = 3e-7; %LIB
    CVS = 4;
    WV = Start:Skip:(Skip*(CVS-1)+Start)
    RegID = 'Liberia';
    kappa = 7/8;
    gamma = 7/6;
case 3
    load Cum_Curve_District_WESTERN_AREA_RURAL.txt;
    tdata_full = Cum_Curve_District_WESTERN_AREA_RURAL(:,1);
    Cdata_full = Cum_Curve_District_WESTERN_AREA_RURAL(:,2);
    N = 500000; %WAF Rural
    Skip = 5;
    Start = 7;
    CVS = 4;
    WV = Start:Skip:(Skip*(CVS-1)+Start)
    lambdaF = 7*1e-7; %WAF Rural
    RegID = 'Western Area Rural';
    kappa = 7/8;
    gamma = 7/6;
case 4
    load Cum_Curve_District_WESTERN_AREA_URBAN.txt;
    tdata_full = Cum_Curve_District_WESTERN_AREA_URBAN(:,1);
    Cdata_full = Cum_Curve_District_WESTERN_AREA_URBAN(:,2);
    N = 1100000; %WAF Urban
    Skip = 5;
    Start = 6;
    CVS = 5;
    WV = Start:Skip:(Skip*(CVS-1)+Start)
    lambdaF = 2.5*1e-6; %WAF Urban
    RegID = 'Western Area Urban';
    kappa = 7/8;
    gamma = 7/6;
case 5
```

```
    load Cum_Curve_District_MONTSERRADO.txt;
    tdata_full = Cum_Curve_District_MONTSERRADO(:,1);
    Cdata_full = Cum_Curve_District_MONTSERRADO(:,2);
    N = 1200000; %MONTSERRADO
    Skip = 2;
    Start = 13;
    CVS = 4;
    WV = Start:Skip:(Skip*(CVS-1)+Start)
    lambdaF = 2*1e-6; %MONTSERRADO
    RegID = 'Montserrado';
    kappa = 7/8;
    gamma = 7/6;
case 6
    load Cum_Curve_District_GUECKEDOU.txt;
    tdata_full = Cum_Curve_District_GUECKEDOU(:,1);
    Cdata_full = Cum_Curve_District_GUECKEDOU(:,2);
    N = 250000; %GUECKEDOU
    Skip = 2;
    Start = 13;
    CVS = 4;
    WV = Start:Skip:(Skip*(CVS-1)+Start)
    lambdaF = 8e-6; %GUECKEDOU
    RegID = 'Gueckedou';
    kappa = 7/8;
    gamma = 7/6;
case 7
    load LonMeas1948.txt;
    tdata_full = LonMeas1948(:,1);
    Cdata_full =  LonMeas1948(:,2);
    N = 8200000; %London
    Skip = 3;
    Start = 8;
    CVS = 4;
    WV = Start:Skip:(Skip*(CVS-1)+Start)
```

```
            lambdaF = 8e-6 %London
            RegID = 'London';
            kappa = 7/8;
            gamma = 7/6;
        case 8
            load SFcumcases1.txt;
            tdata_full = SFcumcases1(:,1);
            Cdata_full =  SFcumcases1(:,2);
            N = 550000; %SF
            Skip = 4;
            Start = 18;
            CVS = 4;
            WV = Start:Skip:(Skip*(CVS-1)+Start)
            lambdaF = 1.2e-5 %SF
            RegID = 'San Fransisco Influenza';
            kappa = 1/2;
            gamma = 1/7;
    end

% Call for Number of Curves
NumCurves = str2double(input('How many curves? ','s'))
    while isnan(NumCurves) || fix(NumCurves) ~= NumCurves || NumCurves<1
        NumCurves = str2double(input('Please enter a positive INTEGER: ', 's')
            );
    end
% Call for Output File Names
FILES = input('Do you wish to create csv files for data generated? (y/n)  ','s
    ');
if strcmpi(FILES,'y')
    Reconstructed_Curves = input('File Name for Reconstructed Curves ','s');
    Reconstructed_Curves_full =strcat(Reconstructed_Curves,'.csv');
    Forecast_early = input('File Name for Early Forecast Curves ','s');
    Forecast_early_full = strcat(Forecast_early,'.csv');
    Poisson_Curves = input('File Name for Poisson Curves ','s');
```

```
    Poisson_Curves_full = strcat(Poisson_Curves, '.csv');
else
    Reconstructed_Curves_full =strcat('x.csv');
    Forecast_early_full = strcat('y.csv');
    Poisson_Curves_full = strcat('z.csv');
end
%-------------------------------------------------------------------------
% Set program values
Cdata_inc_full = [Cdata_full(1,1); diff(Cdata_full(:,1))];
S0 = N;
C0 = Cdata_full(1,1);
E0 = Cdata_inc_full(1,1)/kappa;
I0 = C0;
M = length(tdata_full);
m = 6;
mub = M;
Recon = zeros(mub,4);
FCearly = zeros(mub,4*NumCurves);
ReconPois = zeros(mub,4*NumCurves);
lambdaOrig = lambdaF;
% Iterate through number of early data sets (curves)
for CVnum = 1:4
    lambdaF = lambdaOrig;
    count = 0;
    % Set j to last week
    j = Start+(CVnum-1)*Skip
    count = count + 1
    j
    tdata = tdata_full(1:j,1);
    Cdata = Cdata_full(1:j,1);
    Cdata_inc = Cdata_inc_full(1:j,1);
    n = ceil(2*j);
    %---------------------------------------------------------------------
    % Construct A
```

```
A = zeros(j-1,n+1);
[f,K] = kernel(Cdata_inc, Cdata, tdata);
K_sp = spline(tdata, K);
a = tdata(1); b = tdata_full(mub);
for k1 = 1:n+1
    for k2 = 1:j-1
        A(k2,k1) = integral(@(x) leg(x, k1-1, a, b).*ppval(K_sp,x),...
            tdata(1), tdata(k2+1));
    end
end
% End Construct A

condition = cond(A)
%-------------------------------------------------------------------------
% Subroutine for Regularization Parameter Selection MTSVD
[lambda RD] = RelDisc(A,f,lambdaF,Cdata_inc,n,tdata_full);
RDM(count,1:2) = [lambda, RD]
%-------------------------------------------------------------------------
% Apply Regularization Parameter and Solve MTSVD
[U,S,V] = svd(A);
s = diag(S);
PIS = zeros(j-1,n+1);
for i = 1:j-1
    if s(i) >= lambda
        PIS(i,i) = 1/s(i);
    else
        PIS(i,i) = 1/lambda;
    end
end
PIA = V*(PIS)'*U';
Coef = PIA*f(2:j,1);
%-------------------------------------------------------------------------
% Recover Incidence Curve and Generate Poisson Curves from it
[T, F] = operator(tdata_full,Coef,n);
```

```
F_sp = spline(T, F);
Recon(1:j,CVnum) = F(1:j);
FA = F(1:j);
Cdata = Cdata_full(1:j,1);
FA = Cdata_inc_full(1:j,1);
TFC = T(j+1:end);
timevect = T;
yi = F;
curves = [];
nmb = 1;
for iter = 1:NumCurves
    nmb = nmb + 1   ;
    yirData = zeros(length(yi),1);
    yirData(1) = yi(1);
    for t = 2:length(yi)
        tau = abs(yi(t));
        yirData(t,1) = poissrnd(tau,1,1);
    end
    curves = [curves (yirData)];
end
% Add to Matrix for Poisson Curve Data File
ReconPois(1:j,CVnum*NumCurves-(NumCurves-1):CVnum*NumCurves) = curves(1:j
    ,:);
%-------------------------------------------------------------------------
% Iterate through Number of Curves to Obtain Reconstructed Incidence
nmb = 0;
timevect = linspace(tdata_full(1,1), tdata_full(mub,1), 1000);
BetaExp = zeros(1000, NumCurves);
BetaExpCI = zeros(length(tdata),NumCurves);
FAct = zeros(j,NumCurves);
FFC = zeros(mub,NumCurves);
for iter = 1:NumCurves
    nmb = nmb + 1;
    ExpIncData =curves(1:mub,iter);
```

```
%-------------------------------------------------------------------------
% Construct A
A = zeros(j-1,n+1);
[f,K] = kernel(ExpIncData, cumsum(ExpIncData), tdata);
K_sp = spline(tdata, K);
a = tdata(1); b = tdata_full(mub);
for k1 = 1:n+1
    for k2 = 1:j-1
        A(k2,k1) = integral(@(x) ...
            leg(x, k1-1, a, b).*ppval(K_sp,x), tdata(1), tdata(k2+1));
    end
end
% End Construct A

condition = cond(A);
%-------------------------------------------------------------------------
% Subroutine for Regularization Parameter Selection MTSVD
[lambda RD] = RelDisc(A,f,lambdaF,Cdata_inc,n,tdata_full);
lamV(iter,1) =lambda;
RDisc(iter,1) = RD;
%-------------------------------------------------------------------------
% Apply Regularization Parameter and Solve MTSVD
[U,S,V] = svd(A);
s = diag(S);
PIS = zeros(j-1,n+1);
for i = 1:j-1
    if s(i) >= lambda
        PIS(i,i) = 1/s(i);
    else
        PIS(i,i) = 1/lambda;
    end
end
PIA = V*(PIS)'*U';
Coef = PIA*f(2:j,1);
```

```
        %-------------------------------------------------------------------
        % Recover Incidence Curve and add to Data Matrices

        ybeta = approx(timevect,n,Coef,tdata(1),tdata_full(mub));
        ybetaCI = approx(tdata,n,Coef,tdata(1),tdata_full(mub));
        BetaExp(:, iter) = ybeta;
        BetaExpCI(:, iter) = ybetaCI;
        [T, F] = operator(tdata_full,Coef,n);
        F_sp = spline(T, F);
        FAct(:,iter) = F(1:j);
        FFC(j+1:end,iter) = F(j+1:end);
        FCearly(j+1:end,CVnum*NumCurves-(NumCurves-iter)) = F(j+1:end);
    end  % End Iterations on Number of Curves
end % End Iteration on number of early data sets

% Write fles
csvwrite(Reconstructed_Curves_full,Recon);
csvwrite(Forecast_early_full,FCearly)
csvwrite(Poisson_Curves_full,ReconPois)



end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [f,K] = kernel(Cdata_inc, Cdata, tdata)
global N kappa gamma j C0 E0 I0 S0
K = zeros(j,1);
f = zeros(j,1);

S_p = spline(tdata(1:j,1),Cdata_inc(1:j,1));
```

```
It = zeros(j,1);

for i = 1:j
      It(i,1) = integral(@(t) exp(-gamma*(tdata(i,1) - t)).*ppval(S_p,t),...
          tdata(1), tdata(i));
      K(i,1) = I0*exp(-gamma*(tdata(i,1) - tdata(1,1))) + It(i,1);
      f(i,1) = - log((- Cdata_inc(i,1)/kappa - Cdata(i,1)+ E0 + C0)/S0 + 1);
end

K = K/N;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    function P = leg(x, k, a, b)
        t = (2.*x - a - b)./(b - a);
                if k == 0
                    P1 = 1; P = P1;
                elseif k == 1
                     P2 = t; P = P2;
                else
                        P1 = 1; P2 = t;
                        for i = 2:k
                  P3 = ((2*(i-1)+1).*t.*P2 - (i-1).*P1)./i;
                  P1 = P2; P2 = P3;
                        end
                        P = P3;
                end

    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z = approx(x,m,c,a,b)
```

```
        z = 0;
            for k = 1:m+1
            z = z + c(k).*leg(x, k-1, a, b);
            end
            if z < 0
                z = 0;
            end
        end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function dy = sstm(x,y,Coef,tdata_full,number)
global N kappa gamma mub

dy = zeros(3,1);
dy(1) = -approx(x,number,Coef,tdata_full(1),tdata_full(mub)).*y(1).*y(3)/N;
dy(2) = approx(x,number,Coef,tdata_full(1),...
    tdata_full(mub)).*y(1).*y(3)/N - kappa*y(2);
dy(3) = kappa*y(2) - gamma*y(3);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [T,F] = operator(tdata_full,Coef,number)
global S0 E0 I0 kappa mub

options = odeset('RelTol',1e-4,'AbsTol',1e-6);
[T,Y] = ode23s(@(x,y) sstm(x,y,Coef,tdata_full,number),...
    tdata_full(1:mub,1),[S0 E0 I0], options);

F = kappa*Y(:,2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [lam RD] = RelDisc(A,f,initlam,Cdata_inc,n,tdata_full)
global j
Lmcount = 1;
OP = [];
[U,S,V] = svd(A);
SVS = diag(S);
s = diag(S);
svec = [];
%INITIAL RANGE
svec = linspace(initlam,initlam*1e3,30);
SKP = svec(2)-svec(1);
    for pw = 1:length(svec)
            lambda = svec(pw);
     PIS = zeros(j-1,n+1);
     for i = 1:j-1
         if s(i) >= lambda
             PIS(i,i) = 1/s(i);
         else
             PIS(i,i) = 1/lambda;
         end
     end
     PIA = V*(PIS)'*U';
     Coef = PIA*f(2:j,1);
     [Tt, Ft] = operator(tdata_full,Coef,n);
     RD = norm(Ft(1:j)-Cdata_inc)/norm(Cdata_inc);
     OP(Lmcount,1:2) = [RD lambda];
     Lmcount = Lmcount + 1;
    end
OP
[Val Ix] = min(OP(:,1));
%NEW RANGE
if Ix>1
    if Ix==30
```

```
        svec = linspace(OP(end-1,2),OP(end,2),20);
    else
        svec = linspace(OP(Ix-1,2),OP(Ix+1,2),20);
    end
else
    svec = linspace(OP(1,2),OP(3,2),20);
end

OP=[];
Lmcount = 1;
  for pw = 1:length(svec)
          lambda = svec(pw);
    PIS = zeros(j-1,n+1);
    for i = 1:j-1
        if s(i) >= lambda
            PIS(i,i) = 1/s(i);
        else
            PIS(i,i) = 1/lambda;
        end
    end
    PIA = V*(PIS)'*U';
    Coef = PIA*f(2:j,1);
    [Tt, Ft] = operator(tdata_full,Coef,n);
    RD = norm(Ft(1:j)-Cdata_inc)/norm(Cdata_inc);
    OP(Lmcount,1:2) = [RD lambda];
    Lmcount = Lmcount + 1;
  end
OP
% COMMENT THIS OUT WITH LARGE NUMBER OF CURVES
figure
plot(OP(:,2),OP(:,1))
%Reg Parameter
[RD mnrdix] = min(OP(2:end,1));
lam = OP(mnrdix+1,2);
```

```
end
```

A.5 MATLAB CODE 5 - Plotting the forecasting with early data

```
function beta_line_MTSVD_FC_ST_Uncertainty_Plots
% This Plots results from beta_linear_MTSVD_FC_DP_Uncertainty_GenFiles2
close all
clear all
clc
format long
warning('off','all')

global mub
%-------------------------------------------------------------------------
% USER INPUT
DSet = str2double(input(['Choose from among the following data sets by number
    indicated',...
        '\n 1) Sierra Leone',...
        '\n 2) Liberia',...
        '\n 3) London Measles',...
        '\n 4) San Fransisco Pandemic Influenza',...
        '\n',...
        '\n Which?  '],'s'));
while isnan(DSet) || fix(DSet) ~= DSet || DSet<1 ||DSet>4
    DSet = str2double(input('Please enter and INTEGER between 1 and 4: ', 's')
        );
end
   switch DSet
       case 1
           load SLcumcases.txt;
           tdata_full = SLcumcases(:,1);
           Cdata_full = SLcumcases(:,2);
           curves = csvread('ReconPois_SL.csv');
```

```
        Recon = csvread('Recon_SL.csv');
        FCearly = csvread('FCearly_SL.csv');Skip = 5;
        RegID = 'Sierra Leone';
        MX = 950;
    case 2
        load LIBcumcases_rev.txt;
        tdata_full = LIBcumcases_rev(:,1);
        Cdata_full = LIBcumcases_rev(:,2);
        curves = csvread('ReconPois_LibA.csv');
        Recon = csvread('Recon_LibA.csv');
        FCearly = csvread('FCearly_LibA.csv');
        RegID = 'Liberia';
        MX = 950;
    case 3
        load LonMeas1948.txt;
        tdata_full = LonMeas1948(:,1);
        Cdata_full =  LonMeas1948(:,2);
        curves = csvread('ReconPois_Lon.csv');
        Recon = csvread('Recon_Lon.csv');
        FCearly = csvread('FCearly_Lon.csv');
        RegID = 'London';
        MX = 2300;
    case 4
        load SFcumcases1.txt;
        tdata_full = SFcumcases1(:,1);
        Cdata_full =  SFcumcases1(:,2);
        curves = csvread('ReconPois_SF.csv');
        Recon = csvread('Recon_SF.csv');
        FCearly = csvread('FCearly_SF.csv');
        RegID = 'San Fransisco Influenza';
        MX = 2900;
end
%-------------------------------------------------------------------------
% End User Input
```

```
% Determine Skip and Number of Curves in Data Files
size(curves)
size(Recon)
size(FCearly)
mub = length(curves(:,1));
Cdata_inc = [Cdata_full(1,1); diff(Cdata_full(:,1))];
timevect = (1:mub)';
trig = 0;
Init = 1;
InitWk = find(FCearly(:,1)~=0,1)-1
while trig==0
    if find(FCearly(:,Init)~=0,1)-1~=InitWk
        trig = 1;
        NumCurves = Init-1
    else
        Init = Init + 1;
    end
end
Skip = find(FCearly(:,Init+1)~=0,1) - find(FCearly(:,1)~=0,1)
Iter = length(FCearly(1,:))/NumCurves
%-------------------------------------------------------------------------
%%PLOT

figure1=figure;

axes2 = axes('Parent',figure1,...
    'AmbientLightColor',[0.941176470588235 0.941176470588235
        0.941176470588235]);
box(axes2,'on');
hold(axes2,'all');
hold on
Iter = 4;
MIter = 4;
```

```
for j = 1:Iter
    Wk = InitWk+(j-1)*Skip
    pd = [];
    for i = tdata_full(Wk+1):tdata_full(mub)
        pd=fitdist(FCearly(i,j*NumCurves-(NumCurves-1):j*NumCurves)', 'Normal')
            ;
        mu(i) = pd.mu;
        sig(i) = pd.sigma;
    end
   MXin = max(1.96*sig(1:Wk+Skip) + mu(1:Wk+Skip));
switch j
    case 1
    h1a = plot(tdata_full(1:Wk),Recon(1:Wk,j),'-k','LineWidth',2);
    h4a = errorbar(tdata_full(Wk+1:Wk+Skip),mu(tdata_full(Wk+1:Wk+Skip)),1.96*
        sig(tdata_full(Wk+1:Wk+Skip)),'k');
    h3a = plot(tdata_full(Wk+1:Wk+Skip),mu(tdata_full(Wk+1:Wk+Skip)),':k','
        LineWidth',2);
    case 2
    h1b = plot(tdata_full(1:Wk),Recon(1:Wk,j),'-b','LineWidth',2);
    h4b = errorbar(tdata_full(Wk+1:Wk+Skip),mu(tdata_full(Wk+1:Wk+Skip)),1.96*
        sig(tdata_full(Wk+1:Wk+Skip)),'b');
    h3b = plot(tdata_full(Wk+1:Wk+Skip),mu(tdata_full(Wk+1:Wk+Skip)),':b','
        LineWidth',2);
    case 3
    h1c = plot(tdata_full(1:Wk),Recon(1:Wk,j),'-g','LineWidth',2);
    h4c = errorbar(tdata_full(Wk+1:Wk+Skip),mu(tdata_full(Wk+1:Wk+Skip)),1.96*
        sig(tdata_full(Wk+1:Wk+Skip)),'g');
    h3c = plot(tdata_full(Wk+1:Wk+Skip),mu(tdata_full(Wk+1:Wk+Skip)),':g','
        LineWidth',2);
    case 4
    h1d = plot(tdata_full(1:Wk),Recon(1:Wk,j),'-m','LineWidth',2);
    h4d = errorbar(tdata_full(Wk+1:Wk+Skip),mu(tdata_full(Wk+1:Wk+Skip)),1.96*
        sig(tdata_full(Wk+1:Wk+Skip)),'m');
```

```
    h3d = plot(tdata_full(Wk+1:Wk+Skip),mu(tdata_full(Wk+1:Wk+Skip)),':m','
        LineWidth',2);
end

end
h2 = plot(tdata_full,Cdata_inc,'*r');
switch Iter
    case 1
    legend([h1a h3a h4a h2],{sprintf('Recovered Incidence - %d Wks',Wk),
        sprintf('Mean Value forecast - %d Wks',Wk),sprintf('Confidence
        Intervals - %d Wks',Wk),'Real Data'},...
    'FontSize',12,'Location','best')
    case 2
    legend([h1a h3a h4a h1b h3b h4b h2],{sprintf('Recovered Incidence - %d Wks
        ',Wk-Skip),sprintf('Mean Value forecast - %d Wks',Wk-Skip),sprintf('
        Confidence Intervals - %d Wks',Wk-Skip),...
         sprintf('Recovered Incidence - %d Wks',Wk),sprintf('Mean Value
             forecast - %d Wks',Wk),sprintf('Confidence Intervals - %d Wks',Wk)
             ,'Real Data'},...
    'FontSize',12,'Location','best')
    case 3
    legend([h1a h3a h4a h1b h3b h4b h1c h3c h4c h2],{sprintf('Recovered
        Incidence - %d Wks',Wk-Skip*2),sprintf('Mean Value forecast - %d Wks',
        Wk-Skip*2),sprintf('Confidence Intervals - %d Wks',Wk-Skip*2),...
         sprintf('Recovered Incidence - %d Wks',Wk-Skip),sprintf('Mean Value
             forecast - %d Wks',Wk-Skip),sprintf('Confidence Intervals - %d Wks
             ',Wk-Skip),...
         sprintf('Recovered Incidence - %d Wks',Wk),sprintf('Mean Value
             forecast - %d Wks',Wk),sprintf('Confidence Intervals - %d Wks',Wk)
             ,'Real Data'},...
    'FontSize',12,'Location','best')
    case 4
    legend([h1a h3a h1b h3b h1c h3c h1d h3d h2],{sprintf('Recovered Incidence
        - %d Wks',Wk-Skip*3),sprintf('Mean Value forecast - %d Wks',Wk-Skip*3)
```

```
        ,...
        sprintf('Recovered Incidence - %d Wks',Wk-Skip*2),sprintf('Mean Value
            forecast - %d Wks',Wk-Skip*2),...
        sprintf('Recovered Incidence - %d Wks',Wk-Skip),sprintf('Mean Value
            forecast - %d Wks',Wk-Skip),...
        sprintf('Recovered Incidence - %d Wks',Wk),sprintf('Mean Value
            forecast - %d Wks',Wk),'Real Data'},...
    'FontSize',12,'Location','best')

end
xlabel({'Number of Weeks'},'LineWidth',2,'FontSize',12,'FontName','Computer
    Modern');
ylabel({'Incidence Cases'},'LineWidth',2,'FontSize',12,'FontName','Computer
    Modern');
title(sprintf('Uncertainty in the Reconstruction of Incidence Cases - %s',
    RegID),'FontSize',16)
axis([0 mub 0 MX])
hold off



end
```

Appendix B

DATA SETS - SAMPLE

B.1 Sierra Leone - EVD [1]

Cumulative Case Data from the 2014-15 EVD outbreak in Sierra Leone. Week 1 corresponds to 5/25/14 and Week 66 to 8/23/15. The EVD outbreak was declared ended March 17, 2016 by the World Health Organization (WHO) with a cautionary statement regarding reemergence.

Week	Cumulative Cases
1	3
2	22
3	58
4	131
5	192
6	246
7	339
8	451
9	527
10	632
11	783
12	928
13	1061
14	1244
15	1446
16	1675
17	1920
18	2341
19	2752
20	3164
21	3597
22	4142

23	*4705*
24	*5333*
25	*5969*
26	*6508*
27	*7045*
28	*7695*
29	*8152*
30	*8612*
31	*9026*
32	*9554*
33	*9949*
34	*10281*
35	*10452*
36	*10587*
37	*10706*
38	*10818*
39	*10941*
40	*11043*
41	*11126*
42	*11240*
43	*11300*
44	*11348*
45	*11392*
46	*11426*
47	*11455*
48	*11476*
49	*11490*
50	*11507*
51	*11515*
52	*11538*
53	*11572*
54	*11596*
55	*11621*
56	*11653*

57	*11733*
58	*11801*
59	*11821*
60	*11829*
61	*11838*
62	*11843*
63	*11847*
64	*11849*
65	*11857*
66	*11859*

B.2 London Measles, 1948 [1]

Weekly Cumulative Case Data from the 1948-49 Measles Season in London, England. Week 1 corresponds to 12/25/48 and Week 40 to 9/24/49.

Week	*Cumulative Cases*
1	*110*
2	*404*
3	*722*
4	*1042*
5	*1399*
6	*1886*
7	*2382*
8	*3191*
9	*3968*
10	*4799*
11	*5880*
12	*6914*
13	*8145*
14	*9419*
15	*10906*
16	*12659*
17	*14294*

18	*15993*
19	*17534*
20	*18704*
21	*19812*
22	*20917*
23	*21870*
24	*22953*
25	*23754*
26	*24662*
27	*25275*
28	*25989*
29	*26512*
30	*27002*
31	*27381*
32	*27684*
33	*27957*
34	*28180*
35	*28338*
36	*28458*
37	*28550*
38	*28616*
39	*28658*
40	*28708*